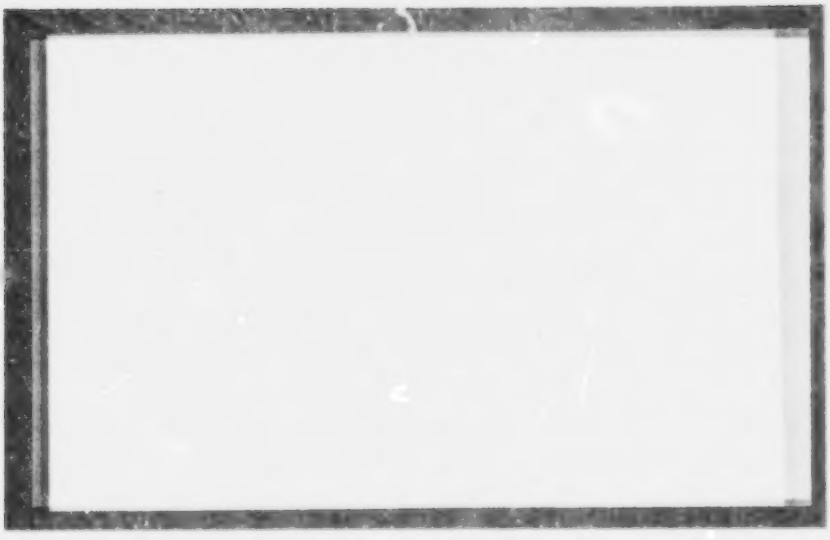
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

UN SYSTÈME INFORMATIQUE POUR LE DÉVELOPPEMENT ET L'EXPÉRIMENTATION D'ALGORITHMES DE GÉNÉRATION DE TOURNÉES

par

Jean-Yves Potvin

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour l'aide et le soutien à la recherche.

Université de Montréal

Centre de recherche sur les transports - Publication #522

Département d'Informatique et de Recherche opérationnelle - Publication #613

Mai 1987

Dépôt légal - 2e trimestre 1987

Table des matières

Sommaire vi

Introduction 1

Chapitre 1. Approches interactives pour la résolution de problèmes en transport 3

1.1 Une vue d'ensemble de la problématique 3

1.2 Les systèmes interactifs d'optimisation 5

1.3 Les systèmes experts 8

1.3.1 Les frames 9

1.3.2 Les systèmes experts et la recherche opérationnelle 12

1.4 Relations de la présente thèse avec les autres travaux 20

1.5 Contribution 21

Chapitre 2. Un système graphique interactif pour la génération de tournées 22

2.1 Description de la classe des problèmes traités 22

2.2 Un système intégré 23

2.3 Les classes d'objets du système 28

2.3.1 La classe RESEAU et ses classes d'appoint 30

2.3.2 La classe NOEUD, ses sous-classes et classes d'appoint 39

2.3.3 La classe ARC 41

2.3.4 La classe RUE 42

2.3.5 La classe VEHICULE 42

2.3.6 La classe ROUTE 43

2.3.7 La classe FENETRE et ses sous-classes 44

2.4 Les bénéfices liés à la programmation orientée objet 44

Chapitre 3. L'heuristique générale: concepts théoriques 47

3.1 Le modèle général 48

3.2 L'opérateur d'initialisation 52

3.2.1 Les points de décision 53

3.3	L'opérateur d'ajout	54
3.3.1	Les points de décision	56
3.3.2	La spécification des contraintes	58
3.4	L'opérateur d'échange	60
3.4.1	Les points de décision	62
3.4.2	Equivalence MTSP - TSP	62
3.5	L'opérateur d'échange: une procédure k-opt étendue	64
3.5.1	La procédure d'échange k-opt*	65
3.5.2	Un algorithme itératif	70
3.5.3	Résultats expérimentaux	71
3.5.4	L'ajout de contraintes	73
3.6	L'opérateur d'échange: une approche multicritères	75
3.7	L'activation des opérateurs	79
Chapitre 4.	L'heuristique générale: concepts pratiques	81
4.1	La classe RESEAU-HEURISTIQUE	81
4.2	Quelques considérations d'ordre pratique	84
4.3	Les facilités offertes pour la définition des formules	85
4.4	Quelques exemples	91
Chapitre 5.	Une application concrète de l'heuristique générale	98
5.1	Présentation du problème	98
5.2	Un algorithme de résolution spécialisé (POST9)	99
5.3	L'heuristique générale	100
5.3.1	Approche séquentielle 1 (SEQ1)	100
5.3.2	Approche séquentielle 2 (SEQ2)	101
5.3.3	Approche parallèle (PAR1)	102
5.4	Conclusion	102
5.5	Observations complémentaires	103

Chapitre 6. Extensions et développements futurs 114

6.1 Les extensions ne modifiant pas la vocation actuelle du système 114

6.1.1 L'introduction de nouveaux opérateurs de base 114

6.1.2 La résolution de problèmes comptant plusieurs origines 115

6.1.3 La résolution de problèmes apparentés 115

6.1.4 La génération automatique de programmes 115

6.1.5 L'implantation de l'heuristique générale sur de nouveaux supports 116

6.2 Les extensions visant à produire un système expert 116

Conclusion 119

Annexe 120

Remerciements 126

Bibliographie 127

Table des illustrations

1.1	Vers un système expert pour la génération de tournées et d'horaires	4
1.2	La représentation du savoir à l'aide de frames	9
1.3	Savoir structuré hiérarchiquement	11
1.4	Organisation hiérarchique d'un plan	14
1.5	Résolution d'un conflit	16
2.1	Deux routes originant du noeud 8	25
2.2	Un menu principal et deux sous-menus	26
2.3	Boucle de rétroaction entre l'usager et le système	28
2.4	Les classes d'objets du système ALTO	29
2.5	Fenêtres d'inspection pour un noeud et un arc	30
2.6	Résultat d'un commis-voyageur	36
2.7	Accès sélectif à un arrêt	37
2.8	Interdiction de virage	42
3.1	Le modèle général	49
3.2	Initialisation des routes R_x, R_y et R_z	52
3.3	Opérateur INIT	53
3.4	Sélection et insertion de l'arrêt n_3	55
3.5	Opérateur AJOUT	55
3.6	Un échange de type 2-opt	61
3.7	Opérateur ECHANGE	61
3.8	Equivalence 2TSP - TSP	63
3.9	Réunification de deux sous-chaînes	65
3.10	Réunification de trois sous-chaînes	66
3.11	Génération d'un tour unique	68
3.12	Tours 2-opt et 2-opt*	69
3.13	Résultats expérimentaux	72
3.14	Echanges 2-opt et 2-opt*	74
3.15	Fonctions d'utilité partielle pour un critère de qualité	76
3.16	Séquences d'activation des opérateurs	79
3.17	Génération de N routes	80

4.1	Le retour arrière	**82**
4.2	Un exemple de "programme"	**83**
4.3	Les notions d'angle et d'angle horaire	**90**
5.1	Accès sélectif à un arrêt	**99**
5.2	La zone Bourassa	104
5.3	Algorithme POST9	105
5.4	Algorithme SEQ1	106
5.5	Algorithme SEQ2	110
5.6	Algorithme PAR1	112

Sommaire

La présente thèse introduit un système informatique ayant pour but de favoriser et accélérer le développement d'algorithmes de génération de tournées pour des véhicules de transport. Ce système repose en particulier sur une heuristique générale que l'usager, en l'occurrence le concepteur d'algorithmes, instancie avec ses propres formules afin d'orienter le processus de résolution vers les solutions désirées. L'heuristique générale apporte ainsi au système une souplesse qui est grandement appréciée lors de la résolution de problèmes pour lesquels il n'existe pas d'algorithmes vraiment satisfaisants. Elle fournit en outre un cadre unificateur pour de nombreux algorithmes classiques décrits dans la littérature, algorithmes qui peuvent dès lors être reproduits à l'aide des formules d'instanciation appropriées.

Le système actuel apporte deux contributions importantes dans le domaine de la génération de tournées. Il fournit tout d'abord, par le biais de l'heuristique générale, un cadre unificateur à l'intérieur duquel il devient possible d'élaborer et spécifier diverses stratégies de résolution destinées à une large classe de problèmes. Enfin, il fournit au concepteur d'algorithmes les moyens d'expérimenter et tester rapidement de nouvelles approches de résolution, favorisant ainsi le développement de l'expertise dans ce domaine

Introduction

De nombreux problèmes liés à la génération de tournées pour des véhicules de transport appartiennent à la classe des problèmes dits "NP-complets" pour lesquels il n'existe pas d'algorithmes exacts susceptibles de les résoudre optimalement en un temps d'ordre polynomial. Cette difficulté a conduit les chercheurs à développer des algorithmes de nature heuristique qui se fondent sur des règles intuitives afin d'accélérer le processus de recherche d'une solution. La qualité de la solution produite à l'aide de ces algorithmes peut toutefois varier fortement selon les problèmes soumis. Cette variabilité s'explique par le manque de connaissances formelles touchant aux stratégies de résolution susceptibles de s'attaquer efficacement à la majorité des problèmes rencontrés. En conséquence, les recherches visant à mettre au point des techniques heuristiques efficaces et robustes afin de résoudre les problèmes complexes rencontrés en pratique en sont encore au stade de l'expérimentation.

Le présent travail introduit donc un système informatique appelé ALTO, présentant des facilités graphiques et interactives, et ayant pour but de favoriser le développement et l'expérimentation d'ALgorithmes de génération de TOurnées. Ce système repose en particulier sur une heuristique générale que l'usager, en l'occurrence le concepteur d'algorithmes, instancie avec ses propres formules afin d'orienter le processus de résolution vers les solutions désirées. L'heuristique générale apporte ainsi au système une souplesse qui est grandement appréciée lors de la résolution de problèmes pour lesquels il n'existe pas d'algorithmes vraiment satisfaisants. Elle fournit en outre un cadre unificateur pour de nombreux algorithmes classiques décrits dans la littérature, algorithmes qui peuvent dès lors être reproduits à l'aide des formules d'instanciation appropriées.

La mise au point d'un tel système se veut un premier pas vers la conception d'un véritable système expert susceptible de s'attaquer à une vaste classe de problèmes liés à la génération de tournées et d'horaires pour des véhicules de transport. A cet effet, certains travaux menés de façon parallèle cherchent à établir une classification des divers types de problèmes rencontrés dans ce domaine. Une telle classification favoriserait ainsi l'identification des caractéristiques majeures des problèmes soumis et, par le biais d'une banque de connaissances expertes intégrée au sein du système, de déduire les stratégies de résolution appropriées.

A cet égard, le système actuel apporte deux contributions importantes. Il fournit tout d'abord, par le biais de l'heuristique générale, un cadre unificateur à l'intérieur duquel il est possible d'élaborer et spécifier diverses approches de résolution destinées à une large classe de

problèmes de génération de tournées. Enfin, il fournit au concepteur d'algorithmes les moyens d'expérimenter et tester rapidement de nouvelles stratégies face aux problèmes rencontrés, favorisant ainsi l'émergence de l'expertise nécessaire à la réalisation d'un véritable système expert.

La thèse elle-même se divise en six chapitres distincts. Le chapitre 1 présente tout d'abord une revue des travaux antérieurs ayant trait à l'utilisation de techniques non entièrement formalisées, laissant une part à l'intuition et au jugement d'un utilisateur expert, afin de résoudre certains problèmes complexes d'optimisation. Le chapitre 2 décrit ensuite l'environnement au sein duquel s'intègre l'heuristique générale. Cet environnement supporte en particulier de nombreuses facilités graphiques et interactives qui simplifient grandement la tâche de l'usager. Les chapitres 3 et 4 s'attachent pour leur part aux concepts théoriques et pratiques à la base de l'heuristique générale. Le chapitre 5 présente ensuite une application concrète dans le domaine de la collecte du courrier, exemple qui permet de démontrer les bénéfices liés à l'utilisation de notre système. Enfin, le chapitre 6 aborde les extensions et améliorations susceptibles d'étendre encore davantage le champ d'application de l'heuristique générale.

Chapitre 1. Approches interactives pour la résolution de problèmes en transport

1.1 Une vue d'ensemble de la problématique

De nombreux problèmes liés à la génération de tournées et d'horaires pour des véhicules de transport présentent une telle complexité qu'il est illusoire pour l'instant d'espérer les résoudre à l'aide d'approches algorithmiques exactes. Ces approches se heurtent en effet à un phénomène d'explosion combinatoire qui limite leur utilisation à des problèmes académiques de petite dimension. Ces difficultés ont donc mené les efforts de recherche vers la conception d'algorithmes heuristiques capables de générer en un temps raisonnable des solutions s'approchant d'une solution optimale. Pour des problèmes simples, tel le problème du commis-voyageur, certaines des heuristiques développées se révèlent performantes et permettent souvent d'obtenir des solutions d'une qualité à peine inférieure à la solution optimale. Dans le cas de problèmes plus complexes, au sein desquels se trouvent des contraintes multiples, les résultats ne sont toutefois pas aussi probants et les stratégies de résolution n'apparaissent pas encore clairement.

L'opportunité d'une intervention humaine est donc évidente. Il s'agit en quelque sorte de marier les capacités de calcul de l'ordinateur aux capacités perceptuelles, intuitives et cognitives de l'usager. Comme nous le verrons à la section 1.2, quelques travaux ont déjà porté sur le bénéfices liés à une intervention humaine lors de la génération d'une solution. Ces réalisations n'autorisent toutefois pas une définition interactive des outils algorithmiques assurant la résolution d'un problème. Tous les algorithmes requis sont déjà codés de manière définitive et seule une modification du code par un programmeur est susceptible d'influencer les approches de résolution.

L'intérêt de notre système provient de ce qu'il s'adresse directement au concepteur d'algorithmes qui est en mesure de modifier interactivement le contenu de l'heuristique "générale". Cette heuristique correspond de fait à un modèle que l'utilisateur instancie avec les formules de son choix afin d'orienter la procédure de résolution. La modélisation de cette heuristique à l'aide de trois opérateurs distincts assurant respectivement l'initialisation des routes, la construction des routes et la modification de la séquence des noeuds à l'interieur des routes permet ainsi de reproduire de nombreux algorithmes classiques décrits dans la littérature tout en autorisant le développement d'approches de résolution entièrement nouvelles.

L'heuristique générale offre donc un cadre unificateur à l'intérieur duquel il devient possible de specifier des stratégies de résolution de natures fort diverses. L'approche

retenue facilite ainsi la conception et l'expérimentation de nouveaux algorithmes heuristiques face aux problèmes rencontrés. Le système actuel représente de fait un premier pas vers la réalisation d'un véritable système expert susceptible de résoudre une classe importante de problèmes de génération de tournées et d'horaires. Comme nous le verrons à la section 1.3, certains problèmes d'optimisation présentant des similitudes avec les problèmes de génération de tournées ont déjà été traités de manière satisfaisante à l'aide d'une approche de type "système expert". Dans notre cas, les connaissances expertes assureraient l'instanciation automatique de l'heuristique générale à partir des caractéristiques des problèmes traités (figure 1.1).

Actuellement, certains travaux menés de façon parallèle cherchent à établir une classification des divers types de problèmes rencontrés dans le domaine de la génération de tournées et d'horaires afin de faciliter leur caractérisation et assurer une instanciation adéquate de l'heuristique générale [8]. Ces travaux couplés à des essais expérimentaux avec le sytème ALTO devraient ainsi favoriser l'émergence de l'expertise nécessaire à l'identification des stratégies de résolution appropriées.

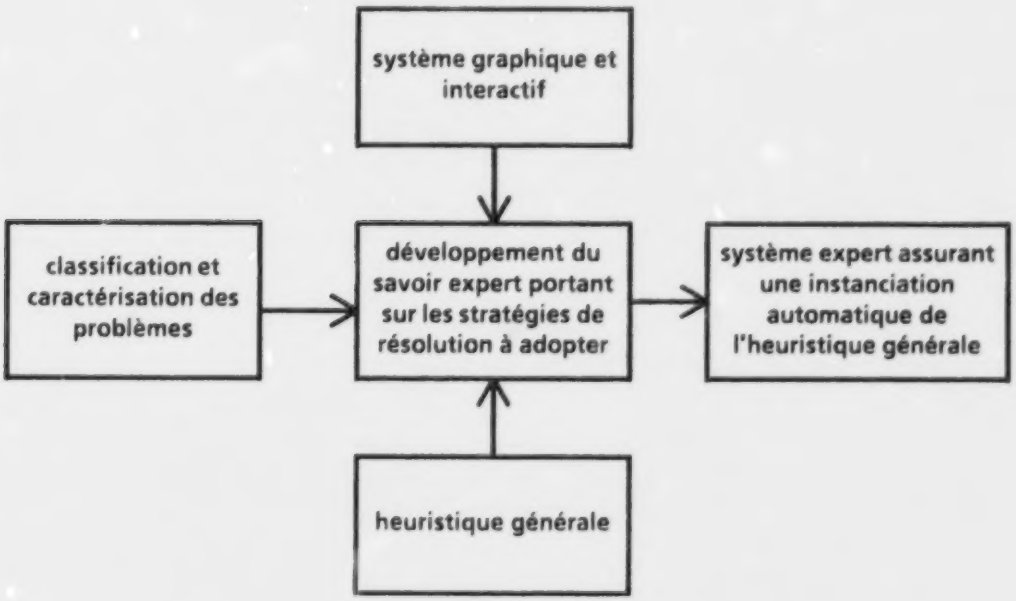


Fig. 1.1 Vers un système expert pour la génération de tournées et d'horaires

Dans la suite de ce chapitre, nous présenterons tout d'abord certains systèmes interactifs d'optimisation offrant à l'usager la possibilité d'intervenir et de modifier les solutions produites. Par la suite, nous aborderons l'étude des systèmes experts qui assurent l'automatisation des processus de décision menant à l'identification d'une solution. Enfin, les similitudes et différences entre le système ALTO et les systèmes précédemment développés seront précisées.

1.2 Les systèmes interactifs d'optimisation

Le principe à la base de tous les systèmes que nous présentons ci-après consiste à favoriser l'intervention de l'utilisateur lors de la résolution d'un problème complexe. On mise ainsi sur l'expérience et l'intuition d'un humain afin de produire des solutions d'une qualité supérieure à celle que peut fournir une approche strictement algorithmique. L'importance de l'intervention humaine peut toutefois varier fortement d'un système à l'autre. Ainsi, l'expert est parfois chargé de générer manuellement les routes désirées et l'ordinateur se borne alors à fournir certaines informations susceptibles de faciliter le travail. A l'opposé, l'intervention humaine peut se limiter à l'initialisation de certains paramètres numériques pour une heuristique particulière. Les deux exemples ci-dessous fournissent justement une illustration de l'apport relatif de l'usager lors de la résolution d'un problème.

a) Un système pour la résolution de problèmes de commis-voyageur [23]

L'approche utilisée par Krolak et Felts cherche à marier la puissance de calcul d'un ordinateur aux capacités de perception et de conceptualisation d'un humain afin de résoudre des problèmes de commis-voyageur de grande dimension. Les étapes menant à l'élaboration d'une solution comprennent:

1) Le traitement des données par ordinateur de façon à présenter à l'usager les caractéristiques majeures du problème à résoudre. Pour ce faire, le système résoud le problème d'affectation que l'on peut dériver de tout problème de commis-voyageur en éliminant les contraintes touchant à la formation d'un tour unique. Ceci génère un ensemble de sous-tours liant les noeuds situés dans une même localité. Ces informations se révèlent fort utiles puisqu'il existe une forte corrélation entre les arcs retenus lors de la résolution du problème d'affectation et les arcs que l'on retrouve au sein de la solution optimale [37].

   La procédure peut être répétée à nouveau en substituant un noeud unique à chacun des sous-tours générés et en résolvant le problème d'affectation correspondant. Cette technique permet ainsi de mettre en évidence des

regroupements locaux et régionaux de noeuds tout en identifiant les arcs les plus susceptibles d'être incorporés au sein d'une "bonne" solution.

2) A l'aide des informations préliminaires fournies par l'ordinateur à l'étape 1, l'utilisateur génère manuellement un tour unique reliant l'ensemble des noeuds.

3) Diverses solutions obtenues à l'aide de certaines heuristiques codées dans le système sont présentées à l'usager à des fins de comparaison avec la solution identifiée à l'étape 2.

4) L'utilisateur construit une seconde solution en y incorporant possiblement les aspects les plus intéressants des solutions présentées à l'étape 3.

5) L'utilisateur peut finalement commander l'application de certains algorithmes d'échange aux portions de la solution qui semblent de qualité douteuse.

Comme on peut le constater, l'usager joue ici un rôle primordial et l'ordinateur se borne à lui fournir les informations susceptibles d'orienter son travail. On cherche donc ici essentiellement à profiter de la puissance de calcul de l'ordinateur afin d'accélérer le travail de l'expert.

Un système plus sophistiqué, basé sur les mêmes principes, a ensuite été développé afin de résoudre des problèmes de génération de tournées pour des véhicules de transport avec contraintes de capacité [24]. Les résultats obtenus furent toutefois moins intéressants car il existe ici une corrélation beaucoup plus faible entre les arcs retenus suite à la résolution du problème d'affectation et les arcs associés à la solution optimale.

b) Un système pour la résolution de problèmes de transport sur demande [5].

Cullen, Jarvis et Ratliff ont développé un système interactif pour la résolution de problèmes en transport sur demande ("dial-a-ride"). L'approche de base repose sur une méthode de partition d'ensemble utilisant une matrice au sein de laquelle chaque colonne représente une route réalisable. L'utilisateur est en mesure de définir cette matrice en introduisant lui-même de nouvelles colonnes (i.e. de nouvelles routes). Le processus de sélection des routes est fondamental puisqu'il est virtuellement impossible de considérer à la fois toutes les routes réalisables: on obtiendrait alors une matrice de dimension beaucoup trop considérable! L'usager doit donc recourir à son expérience et à ses capacités perceptuelles afin d'identifier un bon sous-ensemble de routes, sous-ensemble

qui sera par la suite soumis à l'algorithme de résolution afin d'identifier une solution optimale.

Mettant à profit certains concepts de la théorie de la dualité en programmation linéaire, le système est également en mesure d'informer l'usager sur les bénéfices potentiels liés à l'introduction d'une nouvelle route au sein de la matrice. Ce mécanisme permet ainsi d'orienter l'usager lors du processus de sélection. Il permet également un fonctionnement autonome du système qui est en mesure d'identifier de lui-même les routes présentant un bénéfice maximal.

On voit donc que l'utilisateur se borne ici à définir la matrice qui sera soumise à un algorithme classique pour la résolution de problèmes de partition d'ensemble. Comme nous l'avons mentionné, le système peut même fonctionner de manière entièrement autonome. Ainsi, contrairement à la méthodologie décrite au point (a), l'approche de Cullen, Jarvis et Ratliff privilégie nettement la composante algorithmique du système.

Il existe également de nombreux systèmes qui utilisent les facilités interactives afin d'analyser et améliorer les solutions produites à l'aide d'algorithmes "standards". De tels systèmes regroupent en général trois composantes majeures: une base de données, une procédure de résolution et un module graphique et interactif permettant à l'usager de modifier la base de données ou la solution produite par la procédure de résolution.

C'est selon de tels principes qu'ont été construits, au Centre de Recherche sur les Transports de l'Université de Montréal, les systèmes TRANSCOL [9] et HASTUS [34] utilisés respectivement dans les domaines du transport scolaire et du transport en commun. Le système TRANSCOL permet par exemple de modifier interactivement la solution produite à l'aide de certaines facilités autorisant un transfert manuel des points de demande d'une route à une autre. De telles possibilités sont fort appréciées des utilisateurs puisqu'elles leur permettent de retoucher une solution en fonction des exigences d'une application particulière. Ces systèmes sont toutefois incapables d'assurer une intervention directe de l'usager au niveau même de la procédure de résolution. L'intérêt de notre approche provient de ce que l'usager, en l'occurrence le concepteur d'algorithmes, peut orienter l'heuristique générale à chacune des étapes menant à l'élaboration d'une solution. La souplesse offerte permet ainsi d'évaluer rapidement de multiples approches de résolution face à un problème particulier de façon à identifier celles qui présentent les meilleurs résultats.

Une extension naturelle des systèmes interactifs d'optimisation que nous venons de présenter, consiste à automatiser partiellement ou totalement les processus de décision menant à

l'élaboration d'une solution en y intégrant certaines connaissances acquises par les experts oeuvrant dans le domaine. Les systèmes experts, qui sont l'objet de la prochaine section, apparaissent particulièrement intéressants à cet égard.

1.3 Les systèmes experts (S.E.) [6, 21, 40]

Il existe à l'heure actuelle de nombreux problèmes dont la résolution demande un savoir spécialisé que seuls certains humains possèdent. Ces tâches ne peuvent donc être accomplies que par des experts ayant maîtrisé, par l'étude et l'expérience, le savoir requis. On peut mentionner à titre d'exemple des tâches telles le diagnostic médical, la conception de circuits électroniques ou la réalisation d'expériences biochimiques. Dans cette optique, un système expert est un programme conçu afin d'apporter une aide lors de la réalisation de ces diverses activités. Du point de vue de l'usager, le système se substitue en quelque sorte à un ou plusieurs experts humains susceptibles de trouver une solution satisfaisante au problème. De tels programmes s'avèrent donc fort utiles puisque les experts, en nombre limité, sont souvent peu disponibles.

Les S.E. tels que conçus actuellement favorisent une interaction étroite avec l'usager. Ils se distinguent toutefois des systèmes interactifs classiques par l'utilisation de techniques propres à l'intelligence artificielle, en particulier des techniques pour la représentation des connaissances et l'inférence de faits nouveaux. La base de connaissances s'identifie en quelque sorte au substrat sur lequel agit le système cognitif (ou moteur d'inférence) afin de produire les conclusions désirées. L'un des intérêts majeurs liés à une telle structure provient de ce que la base de connaissances constitue une entité distincte du système cognitif. Il devient donc possible de mettre à jour le contenu de cette base de connaissances tout en conservant l'intégrité du système cognitif.

Au début des années '70, une importance particulière était accordée au système cognitif et l'on cherchait alors à développer un mécanisme d'inférence suffisamment puissant pour être appliqué dans des domaines d'expertise variés. Les chercheurs ont toutefois réalisé assez rapidement qu'un S.E., pour être vraiment utile, se devait de compter sur une base de connaissances importante, spécifique au domaine étudié. Ce savoir, de nature souvent heuristique, a pour but d'orienter le système cognitif vers les avenues de recherche les plus susceptibles de mener à une solution. Les S.E. actuels sont donc munis d'une base de connaissances de dimension considérable et, de ce fait, intègrent des techniques de représentation assurant une gestion efficace de ces connaissances . Dans le cadre du présent travail, nous nous intéresserons essentiellement à la représentation des connaissances sous la forme de "frames", technique retenue pour l'intégration des différents éléments impliqués dans la définition des problèmes de génération de tournées. Cette représentation est ici fort attrayante puisque tout problème de

génération de tournées se décompose aisément en un certain nombre d'entités bien définies, tels les noeuds et les arcs du réseau, les points de demande et les véhicules.

1.3.1 Les frames [30]

Le concept à la base de cette représentation est la modélisation du savoir sous forme d'objets symboliques possédant des attributs et établissant des relations entre eux. Typiquement, un frame permet de décrire diverses classes d'objets possédant des propriétés communes, telles les noeuds dans un réseau ou les humains vivant sur terre. Tous les objets appartenant à une classe donnée possèdent les propriétés de cette classe et la valeur d'une propriété correspond soit à une valeur de défaut définie au sein de la classe, soit à une valeur fournie par l'usager ou pré-calculée pour un objet particulier. Dans le cas où cette propriété est une relation établie avec un autre objet, la valeur associée correspond à un pointeur vers le frame approprié (voir les exemples de frames à la figure 1.2).

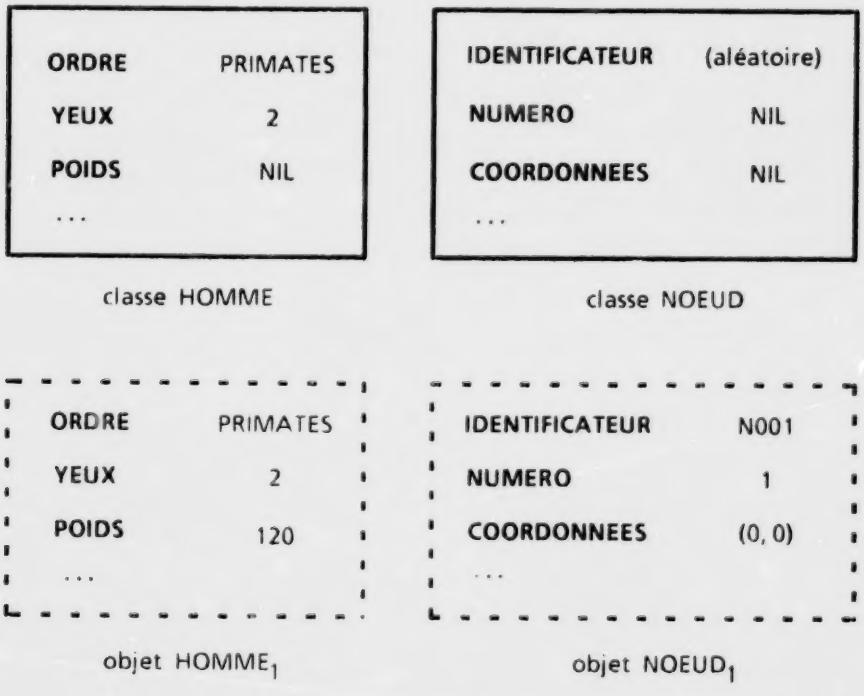
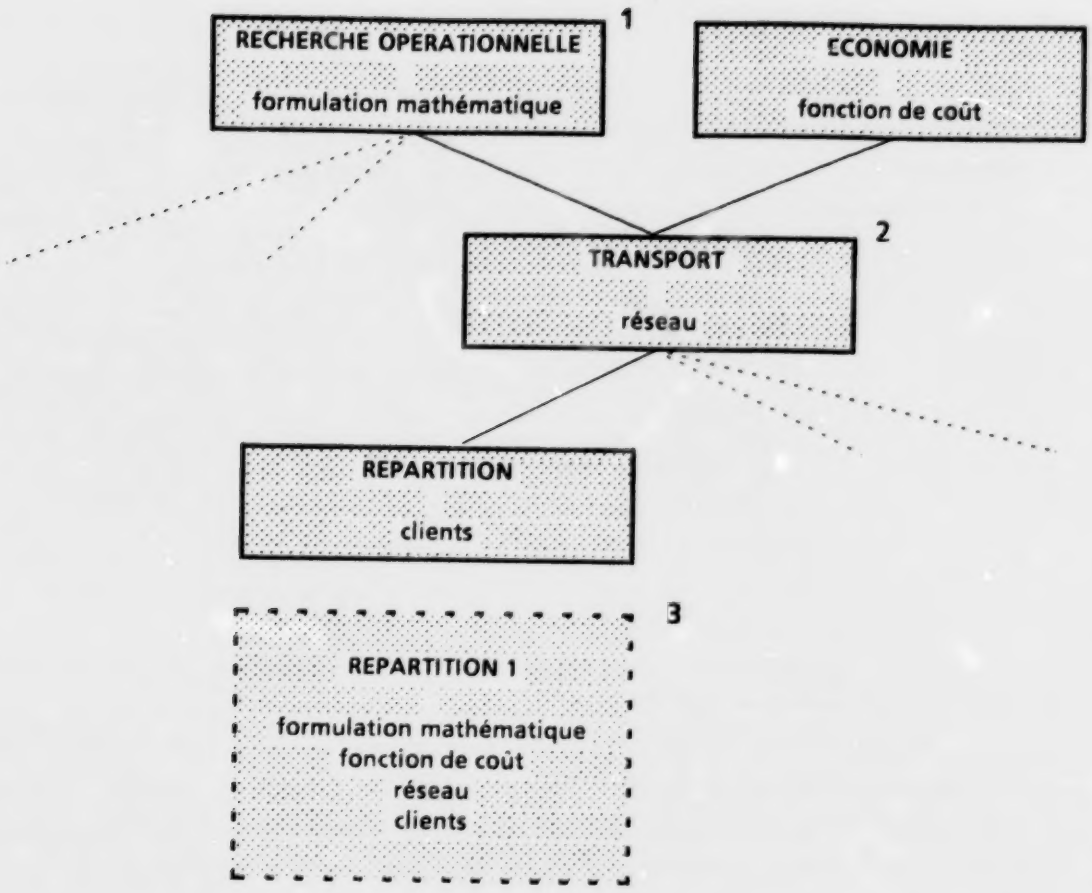


Fig. 1.2 La representation du savoir à l'aide de frames

Comme on le voit, un frame contient un certain nombre de cases ("slots") qui correspondent aux propriétés principales d'une classe d'objets. Lors de la création d'un objet particulier, certaines valeurs sont directement fournies par l'usager (les COORDONNEES de $NOEUD_1$), d'autres réfèrent à des valeurs de défaut présentes au sein de la classe (l'ORDRE de $HOMME_1$). Ajoutons qu'il est possible d'attacher à une case des procédures permettant de calculer une valeur initiale ou d'effectuer une vérification lorsque cette valeur est modifiée. A titre d'exemple, la valeur associée à la case IDENTIFICATEUR de la classe NOEUD correspond à une fonction assurant la génération aléatoire d'un identificateur pour un noeud donné. Le frame réunit ainsi à la fois des concepts touchant à la représentation déclarative et procédurale du savoir [45].

Une notion importante que l'on intègre souvent à la modélisation sous forme de frames est la notion d'héritage. Il est possible en effet de structurer les classes d'objets de manière hiérarchique. Ainsi, un frame représentant une classe spécialisée hérite de toutes les propriétés d'une classe plus générale, située à un niveau plus élevé dans la hiérarchie (figure 1.3).

Les frames sont intéressants dans la mesure où ils permettent de regrouper un ensemble de connaissances communes à certains types d'objets. Ils permettent également de caractériser, par l'intermédiaire des valeurs de défaut, des occurrences typiques d'objets appartenant à une classe donnée. Comme nous le verrons au chapitre suivant, ce formalisme a été mis à profit lors de la représentation symbolique des différents éléments impliqués dans la définition des problèmes de génération de tournées.



1 Frame associé à la classe d'objets RECHERCHE OPERATIONNELLE et regroupant les caractéristiques générales des problèmes rencontrés dans ce domaine (telle une formulation mathématique).

2 Frame associé à la classe d'objets TRANSPORT et regroupant les caractéristiques spécifiques aux problèmes de transport. A noter que le frame TRANSPORT hérite par défaut de toutes les propriétés des classes RECHERCHE OPERATIONNELLE et ECONOMIE.

3 Frame associé à un objet appartenant à la classe REPARTITION. On dit parfois que le frame générique REPARTITION a été instancié afin de produire l'occurrence REPARTITION 1. Cette occurrence possède toutes les propriétés de sa classe et des ancêtres de sa classe.

Fig. 1.3 Savoir structuré hiérarchiquement

1.3.2 Les systèmes experts (S.E.) et la recherche opérationnnelle (R.O.) [8, 31]

De façon générale, il n'existe à l'heure actuelle que peu de liens entre les approches utilisées par les S.E. et les méthodes d'optimisation propres à la R.O. Pourtant, certaines techniques développées en R.O. ont pour but, à l'image des S.E., de fournir un support lors de la prise de décision. Des buts similaires laissent supposer que les deux domaines puissent bénéficier de leurs apports mutuels. En effet:

a) Une approche de type S.E. est en mesure de résoudre des problèmes de R.O. peu formalisés et pour lesquels il n'existe pas d'algorithmes exacts de résolution.

b) Il pourrait être possible d'utiliser certaines techniques d'optimisation propres à la R.O. afin d'accélérer l'obtention d'une solution par un S.E. L'approche heuristique combinée à l'approche mathématique permettrait ainsi d'augmenter la puissance de raisonnement du système.

On ne retrouve pourtant dans la littérature actuelle qu'un nombre restreint de références portant sur des S.E. oeuvrant dans la domaine de la R.O. En outre, seulement deux références mentionnent brièvement l'existence de projets expérimentaux visant à produire des S.E. assurant la génération de routes dans un réseau de transport. Il s'agit tout d'abord d'un système de répartition ("dispatching") qui est en mesure de profiter de l'expérience acquise afin d'enrichir sa base de connaissances [44]. Ce système n'est cependant pas encore opérationnel et nous ne disposons pour l'instant d'aucune information précise à son sujet. La seconde référence touche à la description d'une application particulière du système SNARK [26] qui permet de générer des tournées pour des releveurs de compteurs électriques dans une agglomération urbaine. Cette application, qui compte 44 règles, produit des tournées comptant chacune un nombre d'abonnés situé entre une borne inférieure et une borne supérieure prédéfinies.

Malgré l'absence de travaux touchant directement au domaine étudié, certaines réalisations méritent notre attention car elles s'attaquent à des problèmes présentant des similitudes avec les problèmes de transport. De façon générale, deux sous-disciplines de l'intelligence artificielle ont trouvé des applications en R.O: la première s'intéresse aux problèmes de planification et la seconde à l'analyse et au traitement des contraintes.

La planification

La génération de plans pour l'exécution de tâches diverses présente un intérêt considérable pour les chercheurs en intelligence artificielle. On pense en particulier à la planification des opérations que doit effectuer un robot afin de réaliser une tâche définie. La R.O. s'intéresse également à la génération de plans pour la réalisation de projets complexes tels la construction de maisons ou la production d'objets manufacturés. La planification assure alors un ordonnancement adéquat des tâches nécessaires à l'exécution du projet, tout en permettant une allocation optimale des ressources en temps et matériel. De ce fait, les principaux centres d'intérêt en planification touchent à l'étude des interactions entre les diverses tâches constituantes, à la satisfaction d'objectifs multiples souvent inconciliables, et à la réduction de l'espace de recherche par le biais d'approches hiérarchiques.

Les premiers systèmes développés, tels STRIPS et ABSTRIPS [12, 35], s'intéressaient essentiellement à la planification linéaire où toutes les opérations sont exécutées de façon séquentielle. Ce modèle, relativement simple, convenait bien à la représentation des diverses tâches exécutées par un robot. Par la suite, les chercheurs se sont penchés sur des problèmes plus complexes de planification non linéaire où les tâches sont susceptibles d'être exécutées en parallèle. C'est à ce niveau que l'on trouve d'ailleurs les applications les plus intéressantes dans le domaine de la R.O.

NONLIN [42], issu d'un système plus primitif appelé NOAH [36], fut l'un des premiers systèmes conçus afin de résoudre des problèmes de planification non linéaire. Le système utilise fondamentalement une approche de type hiérarchique assurant le raffinement graduel du plan d'exécution (figure 1.4). NONLIN n'intègre toutefois aucune technique propre à la R.O. et permet simplement la génération d'un plan réalisable: il vérifie que les conditions nécessaires à la réalisation d'une tâche sont satisfaites avant son exécution, corrige les interactions indésirables entre les tâches constituantes et assure un ordonnancement produisant le résultat voulu à la fin du projet.

0 ACTION <DECORATION>

EXPANSION

1 ACTION <POSER LE PLATRE SUR LES MURS>

2 ACTION <POSER LA BASE DU PLANCHER>

3 ACTION <INSTALLATION DU PLANCHER>

4 ACTION <SABLER ET VERNIR LE PLANCHER>

5 ACTION <PEINTURER>

ORDRE 1 → 3; 5 → 4; 2 → 3 → 4;

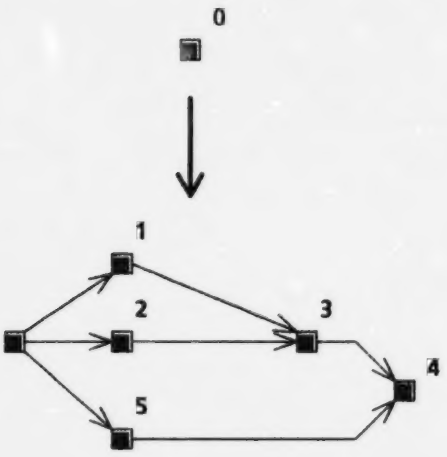


Fig. 1.4 Organisation hiérarchique d'un plan

Les techniques assurant la génération de plans non linéaires ont par la suite été mises à profit afin de résoudre certains problèmes propres à la R.O. A cet égard, deux réalisations nous paraissent particulièrement intéressantes: les systèmes ORDF et NUDGE auxquels nous allons maintenant accorder une attention particulière.

a) ORDF (Ordonnancement à Ressources et Durées Fixées) [32]

Le système ORDF, qui correspond à une instanciation particulière d'un système plus général appelé EAQUE-LRO, s'intéresse à des problèmes d'ordonnancement où la durée des tâches et leurs besoins en ressources sont connus. Contrairement à NONLIN, le système

ORDF intègre certaines techniques propres à la R.O., telles la recherche du chemin critique et la détermination des dates au plus tôt et au plus tard pour chacune des tâches constituant le projet. C'est dire qu'on y retrouve à la fois un savoir déclaratif lié aux différents objets symboliques impliqués dans la définition du problème (tâches, ressources) et un savoir procédural assurant la codification des techniques de R.O. Les objets sont ici représentés sous la forme de frames et les opérateurs permettant de manipuler les objets sont codifiés à l'aide de règles de production.

Dès le départ, la spécification de contraintes de succession pour chacune des tâches définissant un problème assure la génération d'un réseau initial pour le projet. L'analyse du réseau permet ensuite de mettre en lumiere certains conflits au niveau de l'allocation des ressources. De tels conflits surgissent lorsqu'un ensemble de tâches exécutées en parallèle ont des besoins supérieurs à la disponibilité d'une ressource commune.

Ce sont les opérateurs qui permettent au système d'évoluer au sein de l'espace de recherche, en assurant la résolution des conflits à l'aide d'un ordonnancement approprié des tâches conflictuelles (figure 1.5). Pour chacun des nouveaux états genérés, le système évalue les dates au plus tôt et au plus tard de chacune des tâches ainsi que la durée totale du projet. Ceci permet de vérifier que certaines contraintes fondamentales, telle une durée maximale pour le projet, sont respectées. Si tel n'est pas le cas, il existe un mécanisme autorisant soit le retour arrière à une solution réalisable, soit une relaxation de la contrainte appropriée. La dernière alternative est retenue si le coefficient de relaxation de la contrainte (valeur comprise entre 0 et 10) est strictement supérieur à la "note" attribuée à chacun des états pour lesquels un retour arrière est possible (cette note reflète la présence ou l'absence d'alternatives de résolution intéressantes). Le système évolue ainsi au sein de l'espace de recherche jusqu'à l'obtention d'une solution réalisable ou jusqu'à l'épuisement de toutes les alternatives de résolution.

Le système ORDF a ete utilisé pour la génération de plans touchant à la construction de maisons ou de ponts. A notre connaissance, il s'agit actuellement de l'un des seuls S.E. ayant recours à des techniques propres à la R.O. afin d'orienter la recherche vers une solution réalisable.

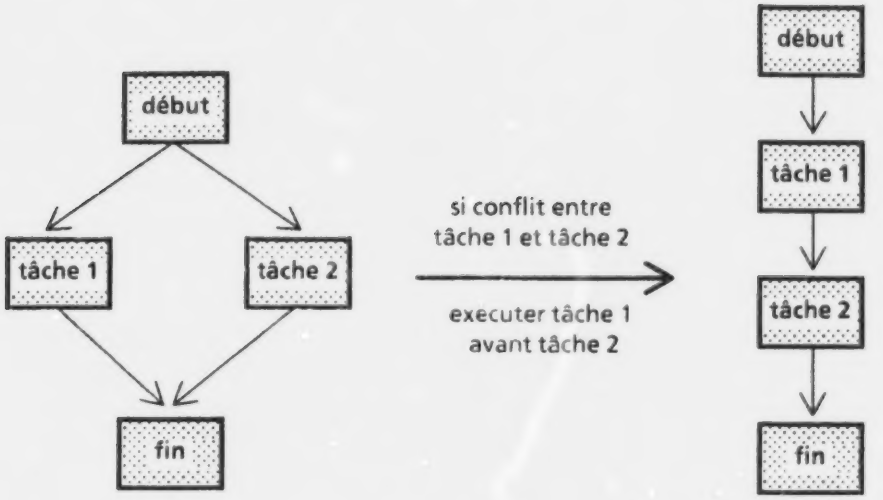


Fig. 1.5 Résolution d'un conflit

b) NUDGE [19, 20]

Le système NUDGE constitue une application intéressante des techniques de planification pour la confection d'horaires de travail dans un bureau ("office scheduling"). Fondamentalement, le système s'apparente à une enveloppe intelligente pour le programme de scheduling BARGAIN. Ainsi, NUDGE est en mesure de transformer une requête informelle et incomplète en une requête formelle assimilable par le programme BARGAIN et ce, grâce à une base de connaissances constituée d'un ensemble de frames structuré hiérarchiquement.

L'intérêt majeur du système provient ici de sa capacité à traiter des objectifs multiples, souvent inconciliables, afin de maximiser la satisfaction de l'usager à l'égard de l'horaire produit. Pour ce faire, les contraintes sont divisées en deux classes distinctes: les prérequis qui s'identifient à des contraintes inélastiques devant être satisfaites de façon prioritaire et les préférences pour lesquelles il existe des alternatives valables.

La stratégie de résolution globale repose sur une approche de type "fix and debug" où un plan est créé en allouant un intervalle de temps à chacune des activités grâce aux valeurs de défaut, préférences ou prerequis inclus au sein de la base de connaissances. Par

la suite, les conflits sont résolus à l'aide de stratégies permettant de satisfaire un nombre maximal de préférences tout en respectant les prérequis (en autant que cela soit possible!). Ces stratégies se divisent en deux catégories: les approches orientées en fonction des ressources (temps et personnel) qui cherchent à relaxer les préférences et les approches orientées en fonction des objectifs qui tiennent en compte le but véritable d'une activité. A titre d'exemple, le fait de déplacer le moment d'une activité pour laquelle seule une préférence est exprimée, s'identifie à une approche de résolution du premier type. Par contre, le fait de remplacer une personne non disponible par une autre personne qualifiée lors de la tenue d'une réunion portant sur un sujet particulier s'identifie à une approche du second type.

Face à un conflit, toutes les stratégies de résolution qui s'appliquent à la situation courante sont activées et la solution assurant la satisfaction du plus grand nombre de préférences est retenue. En cas d'impasse, le système peut soit revenir à un état antérieur, soit relaxer certains prérequis si l'usager en fournit l'autorisation.

De façon générale, il apparaît que l'intégration du programme BARGAIN au sein d'un système faisant appel à une base de connaissances permet d'améliorer la qualité des horaires produits en multipliant les alternatives de résolution face à un problème donné.

L'analyse et le traitement des contraintes

L'analyse de contraintes, domaine de recherche relativement nouveau en intelligence artificielle, ne peut se dissocier entièrement des travaux touchant à la planification. En effet, c'est au cours du développement de systèmes tels MOLGEN et ISIS qu'est apparu la nécessité de traiter adéquatement les contraintes présentes au sein des problèmes étudiés. Ainsi MOLGEN [38, 39], conçu pour la planification d'expériences biochimiques, partitionne un problème en plusieurs sous-problèmes de façon à ce que les contraintes restreignant les valeurs des variables soient propagées d'un sous-problème à l'autre. Dès lors, chaque variable se trouve à accumuler plusieurs contraintes de nature diverse avant qu'une valeur lui soit véritablement affectée. Cette approche permet ainsi de tenir compte des interactions possibles entre les sous-problèmes.

Le système ALICE [25] utilise pour sa part certaines techniques algébriques et heuristiques afin de résoudre des problèmes combinatoires simples, en particulier des problèmes de partition et de recouvrement d'ensemble. Sa tâche s'apparente à celle que l'on rencontre en programmation linéaire ou non linéaire: étant donné un ensemble d'équations restreignant le domaine des variables, est-il possible d'affecter à chaque variable une valeur produisant une

solution réalisable et optimale? Ici, plutôt que d'utiliser la force brute d'un algorithme bien défini, tel l'algorithme du simplexe, ALICE utilise les contraintes afin de réduire progressivement l'ensemble des valeurs accessibles aux différentes variables. La stratégie de résolution s'apparente à une approche de type "hill-climbing": une solution réalisable est d'abord générée de façon progressive à l'aide d'une évaluation de l'importance relative des contraintes à satisfaire, puis certaines transformations de nature heuristique assurent l'identification d'une solution optimale. Le système ALICE est intéressant dans la mesure où il met en évidence la pertinence d'une analyse soignée des contraintes afin de restreindre l'espace de recherche et contrôler l'explosion combinatoire.

Cette idée a été poussé plus avant par Mark S. Fox pour le système ISIS. Comme il s'agit ici d'un S.E. opérationnel évoluant dans un contexte industriel, il vaut la peine de s'y attarder quelque peu.

a) ISIS [14, 15]

Le système ISIS assure le séquencement adéquat des opérations lors de la confection de pièces en usine (problème d'ordonnancement prévisionnel d'atelier ou "job-shop scheduling"). Il s'agit d'un exemple unique où les contraintes sont utilisées explicitement afin de réduire l'espace de recherche et orienter le mécanisme d'inférence vers une solution satisfaisante. Puisque le problème se caractérise par la présence d'une multitude de contraintes variées, ISIS utilise des schémas (frames) pour l'intégration des informations concernant la nature des contraintes et les relations qu'elles établissent entre elles. On y retrouve par exemple le type et l'importance de la contrainte, les diverses relaxations permises et enfin les relations établies avec d'autres contraintes (la satisfaction d'une contrainte particulière peut soit augmenter, soit réduire les alternatives assurant la satisfaction d'une autre contrainte). Toutes ces informations sont essentielles car la stratégie de résolution est entièrement orientée en fonction des contraintes définissant le problème.

De façon générale, l'espace de recherche se divise en 4 niveaux organisés de manière hiérarchique:

i) le niveau 1 assure la selection d'une commande particulière (requête pour la confection d'une pièce).

ii) le niveau 2 procède à une analyse de capacité. Les opérations menant à la confection de la pièce sont ordonnancées et des intervalles de temps leur sont

alloués. Ces intervalles de temps s'identifient à de nouvelles contraintes qui seront propagées au niveau 3.

iii) le niveau 3 procède à l'analyse des ressources. Les machines et autres ressources nécessaires à l'exécution des diverses opérations sont identifiées et des périodes de disponibilité leur sont associées (intervalle de temps au cours duquel la ressource est disponible pour une opération donnée).

iv) le niveau 4 réalise l'affectation définitive des ressources aux diverses opérations de façon à assurer la satisfaction d'un nombre maximal de contraintes.

A l'intérieur de cette structure, les contraintes sont propagées du niveau supérieur (niveau 1) vers les niveaux inférieurs. A chacun des niveaux, une analyse pré-recherche permet de sélectionner les opérateurs chargés d'assurer la génération de nouveaux états au sein de l'espace de recherche. La recherche elle-même retient à chaque itération les "n" états les mieux cotés ("beam search"). Une particularité intéressante du système provient de ce que les contraintes sont explicitement utilisées afin d'évaluer la cote d'un état. Chaque contrainte associé à un état retourne une valeur comprise entre 0 et 2 et c'est la moyenne de ces valeurs qui est retenue. La cote minimale correspond à un état ne satisfaisant aucune contrainte et la cote maximale correspond à un état vérifiant toutes les contraintes sans qu'aucune relaxation ne soit nécessaire. Finalement, une analyse post-recherche évalue la pertinence des solutions proposées en regard des contraintes présentes. Si aucune solution n'apparaît vraiment acceptable, les relations établies entre les diverses contraintes permettent d'identifier certaines relaxations susceptibles de générer de nouvelles alternatives. Il est également possible de "recommencer à neuf" au niveau de l'analyse pré-recherche en ajoutant certains opérateurs précédemment exclus.

En pratique, le système ISIS a permis d'améliorer la productivité dans une usine fabriquant certaines composantes de turbines électriques. L'approche préconisée par ce système met ainsi en relief les bénéfices liés à une utilisation judicieuse des contraintes lors de la recherche d'une solution.

Un autre système expert, appelé OPAL, évolue également dans le domaine de l'ordonnancement prévisionnel d'atelier [2]. Ce système, actuellement en cours de développement, est conçu de façon à établir une nette distinction entre le savoir générique propre à tout problème d'ordonnancement et le savoir spécifique lié à une application particulière. Le savoir générique est intégré au sein d'un module d'analyse de contraintes qui détermine les effets des contraintes de séquence sur les solutions en cours

d'élaboration. Le savoir spécifique correspond par ailleurs à un ensemble de règles expertes permettant d'établir certaines priorités entre les tâches à exécuter. L'intérêt majeur de ce système provient évidemment de cette distinction entre le savoir générique et le savoir spécifique qui permettra de limiter les ajustement nécessaires afin de rencontrer les exigences d'une application particulière.

Enfin, d'autres travaux plus théoriques ont porté sur la représentation des contraintes, non sous la forme binaire classique (satisfaite / non satisfaite), mais sous une forme continue. Ces concepts ont été utilisés en particulier lors de la réalisation d'un système assurant l'établissement d'horaires pour des trains [16]. Le système utilise une technique dite des horaires flous ("fuzzy times") afin de représenter les contraintes présentes au sein d'un problème. Un horaire flou correspond de fait à un ensemble de fenêtres de temps pour le départ et l'arrivée des trains et l'étendue de ces fenêtres varie en fonction des contraintes présentes. Il faut toutefois noter qu'il s'agit d'une application relativement simple puisqu'un train ne dispose que d'un nombre très limité de routes entre deux villes et les ressources utilisées se limitent à la voie ferrée et aux stations.

1.4 Relations de la présente thèse avec les autres travaux

Comme nous l'avons mentionné, le présent travail constitue un premier pas vers la réalisation d'un S.E. oeuvrant dans le domaine de la génération de tournées et d'horaires pour des véhicules de transport. Actuellement, l'expertise qui s'exprime sous la forme de formules assurant l'instanciation de l'heuristique générale est entièrement fournie par le concepteur d'algorithmes. La prochaine phase de développement verra donc à intégrer les connaissances nécessaires à une instanciation automatique de l'heuristique générale en fonction des caractéristiques des problèmes traités. Nous disposerons alors d'un véritable S.E. susceptible de se substituer en tout ou en partie au concepteur d'algorithmes.

Il reste néanmoins que le système, dans sa forme actuelle, représente un outil fort utile pour la résolution de problèmes complexes (cf. chapitre 5). Il offre en particulier un cadre général à l'intérieur duquel il devient possible de définir et tester de nouvelles approches de résolution. De ce fait, il se distingue des systèmes interactifs usuels en offrant à l'usager la possibilité de construire ses propres algorithmes. L'expertise acquise de cette façon devrait ainsi permettre d'accélérer le processus de création du S.E. envisagé.

Ajoutons finalement que le recours au formalisme des frames pour la représentation symbolique des éléments définissant un problème facilite grandement le traitement algorithmique en regroupant les informations apparentées. Ainsi, certaines

caractéristiques d'une route telles la longueur, la demande totale et le temps de parcours sont modifiées lors de l'insertion d'un nouveau point de demande. Le processus de mise-à-jour se trouve alors facilité du fait que toutes les informations pertinentes sont regroupées au sein d'un frame unique décrivant cette route. En outre, le recours à l'approche "orientée objet" ne peut que favoriser l'intégration ultérieure du savoir expert (cf. les bénéfices liés à une intégration du savoir, codé sous forme de règles de production, au sein des objets symboliques [1]).

En conclusion, le système actuel se veut un outil sophistiqué pour le développement et l'expérimentation d'algorithmes de résolution mieux adaptés aux problèmes de génération de tournées. Un tel système pourrait éventuellement servir de base à la conception d'un véritable S.E.

1.5 Contribution

La contribution de cette thèse dans la domaine de la génération de tournées pour des véhicules de transport se situe à trois niveaux:

a) La conception d'une heuristique générale qui permet d'unifier une large classe d'algorithmes classiques décrits dans la littérature. Cette heuristique s'identifiant à un modèle que le concepteur d'algorithmes instancie avec ses propres formules, il devient dès lors possible de concevoir et expérimenter de nouvelles approches de résolution face aux problèmes rencontrés.

b) Le développement d'un environnement sophistiqué supportant une approche de résolution dynamique, où une solution se construit de façon progressive à l'aide des facilités graphiques et interactives qui sont offertes.

c) La mise au point d'un système permettant de développer l'expertise nécessaire à la réalisation d'un véritable système expert.

Les chapitres qui suivent nous permettront de mettre en relief tous les points mentionnés ci-haut. Ainsi, le chapitre 2 étudiera l'environnement graphique et interactif offert par le système ALTO, environnement qui facilite grandement la tâche de l'usager. Les chapitres 3 et 4 s'attacheront pour leur part à une description précise des concepts à la base de l'heuristique générale. Enfin, le chapitre 5 présentera une application concrète dans le domaine de la collecte du courrier par des véhicules postaux.

Chapitre 2. Un système graphique interactif pour la génération de tournées

2.1 Description de la classe des problèmes traités

Les problèmes de génération de tournées s'identifient à une classe très vaste de problèmes présentant des caractéristiques variées. En particulier, les contraintes présentes sont souvent liées aux besoins d'une application spécifique: limitations diverses touchant les véhicules disponibles, contraintes de précédence entre les points de demande, fenêtres de temps, etc.

Pour les besoins de notre recherche, nous avons dû évidemment restreindre cet ensemble. Plus précisément, l'heuristique générale est en mesure de résoudre des problèmes de cueillette ("pick-up") présentant les caractéristiques suivantes:

a) la présence d'une origine (dépôt) unique.

b) la présence d'une flotte de véhicules homogène ou non homogène regroupée au sein de l'origine et destinée à desservir l'ensemble des arrêts (points de demande).

c) la présence d'une demande déterministe à chacun des arrêts. Cette demande doit être recueillie en totalité par le véhicule desservant l'arrêt.

d) la génération de routes bouclées.

e) la présence de l'une ou l'autre des contraintes suivantes:

   i) contraintes de capacité des véhicules

   ii) contraintes d'autonomie des véhicules en terme de distance maximale ou temps de parcours maximal autorisé

   iii) fenêtres de temps au niveau des arrêts

f) la minimisation des coûts d'opération: à cet effet, un certain nombre de critères de qualité peuvent être introduits afin d'évaluer et modifier les solutions obtenues. Ces critères sont:

   i) la longueur des routes

   ii) la compacité des routes

   iii) les temps de passage au niveau des arrêts

   iv) l'utilisation appropriée des véhicules en termes de saturation de la capacité, de la distance maximale ou du temps de parcours maximal.

Les critères retenus nous permettent ainsi d'identifier une classe de problèmes assez générale et représentative de l'ensemble des problèmes liés à la génération de tournées pour des véhicules de transport. Il faut toutefois observer que des extensions à ce modèle sont possibles sans qu'il n'y ait lieu d'altérer les principes gouvernant l'heurisitique générale. C'est le cas en particulier si l'on ajoute de nouvelles contraintes portant sur les véhicules ou les arrêts impliqués dans la définition du problème. Nous pensons ici à des contraintes telles: une capacité maximale pour les véhicules en terme de poids et volume, une compartimentation des véhicules pour le transport de commodités diverses, des contraintes d'exclusion entre les arrêts (deux arrêts donnés ne devant pas appartenir à une même route), etc. Evidemment, la considération explicite de ces contraintes demanderait certains ajustements au niveau de l'implantation effective de l'heuristique générale.

Notre système a donc été conçu afin de faciliter la résolution de tels problèmes. De façon typique, l'usager intègre d'abord à l'environnement un réseau de transport comprenant un certain nombre de noeuds, d'arcs et de rues. Ce réseau est obtenu par lecture d'un fichier d'entrée ou par construction interactive. Dès lors, il devient possible de créer et gérer une flotte de véhicules, de spécifier un ensemble d'arrêts à visiter et de définir une origine au sein de laquelle seront stockés les véhicules. Par la suite, l'usager est en mesure de générer des tournées pour chacun des véhicules à l'aide des algorithmes codés dans le système.

L'heuristique générale, qui fera l'objet des chapitres 3 et 4, représente l'outil fondamental assurant la génération de tournées pour des véhicules de transport. A cette heuristique s'ajoutent deux algorithmes classiques d'optimisation , soit l'algorithme de Dijkstra pour la recherche des chemins plus courts et un algorithme exact de commis-voyageur utilisant une approche de type "branch-and-bound". Ces algorithmes sont à la disposition de l'usager qui peut en commander l'application au moment jugé opportun. L'heuristique générale est également en mesure de les activer automatiquement: l'algorithme de Dijkstra sera appelé si les valeurs désirées n'ont pas déjà été calculées tandis que l'appel à l'algorithme exact de commis-voyageur peut être intégré au sein de l'opérateur d'échange de l'heuristique générale. Bien évidemment, d'autres algorithmes d'optimisation de ce type pourront éventuellement être ajoutés au système afin d'en augmenter encore les possibilités (tel un algorithme d'affectation généralisée permettant de reproduire l'approche de Fisher-Jaikumar [13]).

2.2 Un système intégré

Le programme ALTO, qui constitue le noyau du système graphique et interactif que nous avons conçu, a été développé à l'aide du langage orienté objet LOOPS et implanté sur la

machine "Lisp" Xerox 1108 [41, 43]. Compte tenu du cadre de notre recherche, cette machine présente de nombreux avantages parmi lesquels nous retrouvons:

a) Le support de l'approche "orientée objet"

i) Le langage LOOPS permet de représenter les éléments impliqués dans la définition d'un problème - tels le réseau, les points de demande, les routes et les véhicules - sous la forme d'objets symboliques présentant des caractéristiques ou attributs divers. Cette approche autorise l'inspection et la modification des objets sur une base individuelle. De ce fait, chaque objet s'identifie à une petite banque d'informations directement accessible à l'usager.

ii) L'organisation hiérarchique du savoir permet la spécialisation des classes d'objets en sous-classes plus spécifiques. A titre d'exemple, la classe NOEUD se subdivise en sous-classes afin de rendre compte de certains noeuds "spéciaux" demandant un traitement particulier, tels le dépôt central et les points de demande.

b) La richesse de l'environnement INTERLISP-D

L'environnement INTERLISP-D de la machine "Lisp" offre les avantages suivants:

i) la présence de nombreuses facilités pour la représentation graphique et la manipulation interactive de la structure du réseau

ii) la présence de fenêtres pour l'affichage du réseau, l'interaction avec l'usager et l'inspection des objets symboliques.

iii) le recours aux menus pour la sélection des facilités offertes par le système.

iv) l'utilisation d'une "souris" afin de selectionner un objet particulier (en pointant directement sur celui-ci).

Cet environnement sophistiqué nous a donc permis de produire un système intégrant diverses facilités pour la représentation graphique du réseau, la définition interactive des problèmes et la génération de tournées. Un tel système permet à l'usager d'évoluer au sein d'un environnement très dynamique lors du processus de résolution. Parmi les facilités offertes, mentionnons:

a) les facilités graphiques

Le logiciel GRAPHER [46] a été utilisé afin de réaliser l'implantation des principales fonctions graphiques offertes par le système. Ce logiciel assure l'affichage du réseau à l'intérieur d'une fenêtre et la manipulation interactive de ce réseau (en particulier l'ajout et le retrait de noeuds et d'arcs). Comme il sera précisé lors de la description de la classe RESEAU-AFFICHAGE, l'apparence du réseau est susceptible d'être modifiée selon les goûts et besoins de l'usager. Une telle représentation autorise de plus la visualisation directe des routes générées par les différents algorithmes disponibles (figure 2.1).

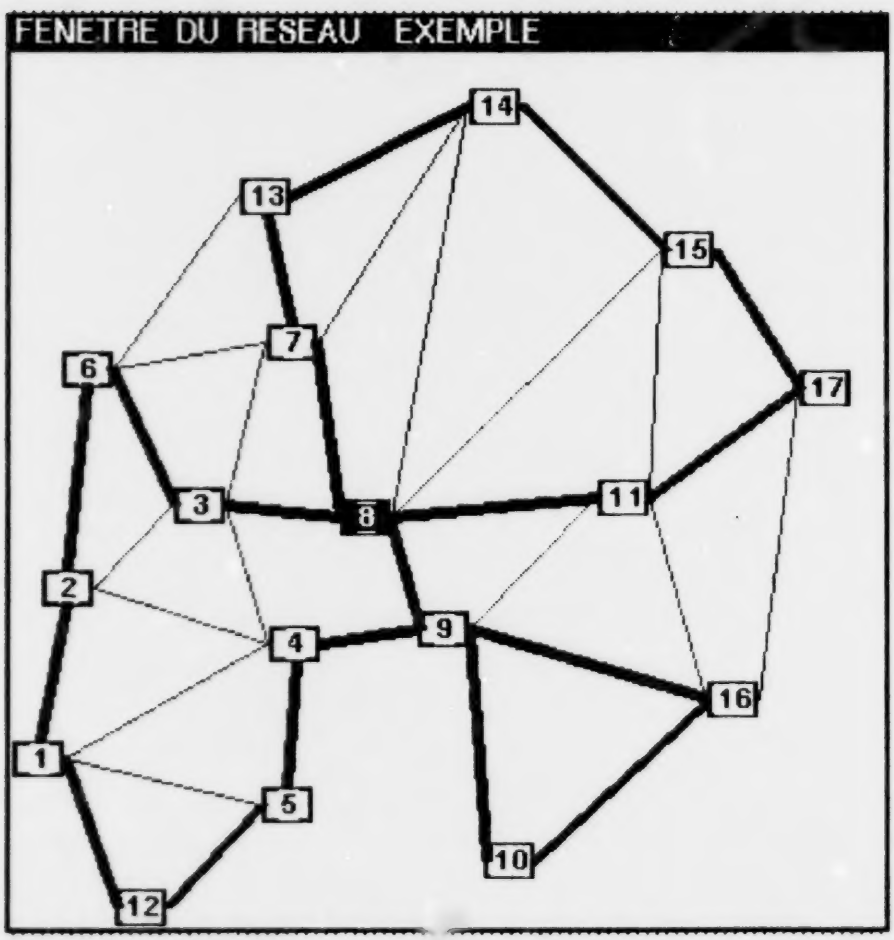


Fig. 2.1 Deux routes originant du noeud 8

b) les facilités interactives

Le système est conçu de façon à permettre à l'usager de quérir ou modifier interactivement les informations touchant à la définition du problème à résoudre. Toutes les facilités interactives sont accessibles par l'intermédiaire de sélections faites dans des menus et les fonctions disponibles dépendent en général du type d'objet que l'on veut traiter.

Pour un arrêt, on retrouve par exemple les menus suivants:

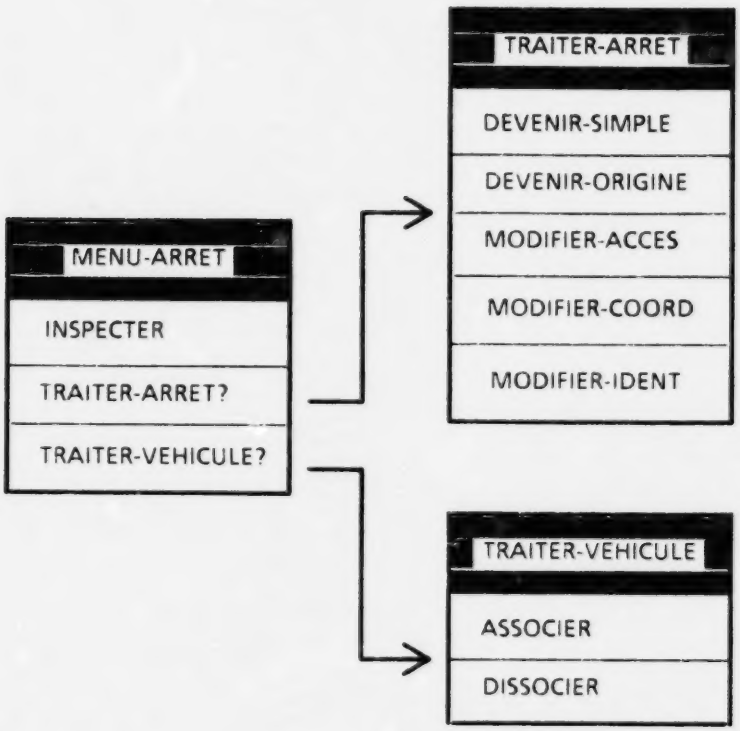


Fig. 2.2 Un menu principal et deux sous-menus

Comme on le voit, la sélection d'un item se terminant par un point d'interrogation entraîne l'activation d'un sous-menu plus spécialisé. Cette organisation hiérarchique facilite le travail de l'usager en regroupant les fonctions apparentées. Les principaux types de menus disponibles sont:

i) les menus associés aux noeuds

Un menu apparaît lorsque l'utilisateur actionne le bouton gauche de la souris à l'intérieur d'un noeud particulier. Les fonctions disponibles diffèrent selon la classe à laquelle appartient le noeud (cf. la description des classes ARRET, ORIGINE et NOEUDSIMPLE à la section 2.3).

ii) les menus associés aux arcs

Si l'usager actionne le bouton centre de la souris à l'intérieur d'un noeud particulier, un menu des arcs incidents à ce noeud apparaît aussitôt. La sélection d'un arc particulier entraîne alors l'activation d'un menu spécialisé pour le traitement de l'arc (cf. la description de la classe ARC à la section 2.3).

iii) le menu associé à la fenêtre du réseau

Un menu apparaît lorsque l'usager actionne la souris au niveau de la bordure supérieure de la fenêtre contenant le réseau. Ce menu assure l'activation de fonctions générales s'appliquant à l'ensemble des noeuds, des arcs ou des véhicules.

iv) le menu des rues et des véhicules

Le système gère deux autres menus supplémentaires: un menu regroupant l'ensemble des rues définies au sein du réseau et un menu regroupant l'ensemble des véhicules disponibles. Ces menus sont automatiquement activés lorsque l'usager sélectionne une fonction requérant la spécification d'une rue ou d'un véhicule particulier.

c) la définition d'un problème

L'utilisateur dispose de toutes les facilités lui permettant de générer interactivement le problème qu'il désire. En l'occurrence, les différents menus offrent les possibilités suivantes: la modification de la structure du réseau incluant l'ajout ou le retrait de noeuds et d'arcs, la relocalisation de l'origine, la mise-à-jour de l'ensemble des arrêts et des véhicules et la modification de la valeur de certains attributs tels la demande à un arrêt ou la capacité d'un véhicule.

d) la génération de tournées

Comme nous l'avons mentionné précédemment, un certain nombre de procédures assurant la génération de parcours sont présentes dans le système. L'usager est ainsi en mesure d'activer ces algorithmes et d'en observer les résultats. Dès lors, il devient possible de simuler une boucle de rétroaction où l'usager modifie les paramètres de résolution du problème en fonction des résultats obtenus (figure 2.3). Toute la conception de l'heuristique générale repose d'ailleurs sur un tel principe.

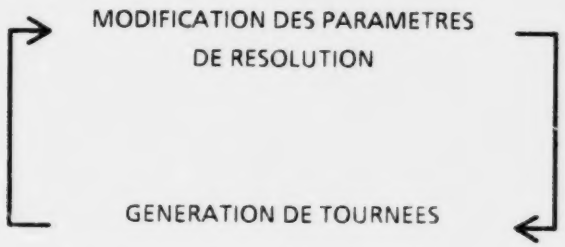


Fig. 2.3 Boucle de rétroaction entre l'usager et le système

2.3 Les classes d'objets du système

L'organisation hiérarchique des classes d'objets utilisées par le système est presentée à la figure 2.4. La classe primitive OBJET regroupe les sous-classes: RESEAU, NOEUD, ARC, RUE, VEHICULE et ROUTE. Par ailleurs, la sous-classe NOEUD se divise elle-même en deux sous-classes spécialisées: NOEUDSIMPLE et NOEUDSPECIAL. Enfin un NOEUDSPECIAL est soit un ARRET soit une ORIGINE. Le système contient également un ensemble de classes d'appoint, connues en LOOPS sous le nom de "mixins", qui ajoutent à une classe certaines propriétés supplémentaires. RESEAU-AFFICHAGE par exemple, contient toutes les informations essentielles à l'affichage du réseau à l'intérieur d'une fenêtre. Enfin, un certain nombre de classes assurent une gestion adéquate des fenêtres au sein de l'environnement INTERLISP-D: il s'agit de la classe FENETRE et des sous-classes FENETRE-RESEAU, FENETRE-INTERACTIVE et FENETRE-RESULTATS.

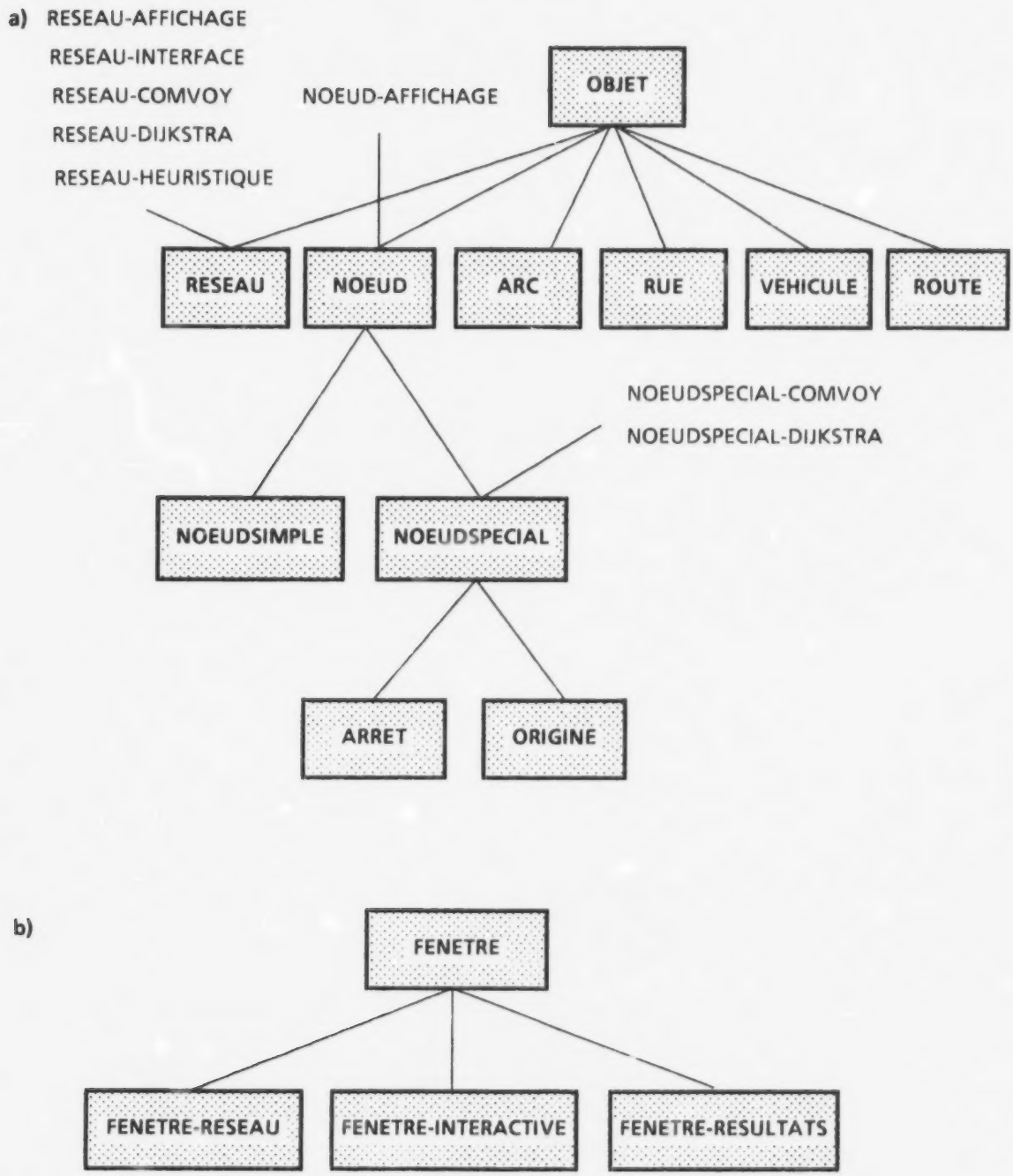


Fig. 2.4 Les classes d'objets du système ALTO

Il faut remarquer que chacune des classes d'objets définies en LOOPS regroupe un ensemble d'attributs et de méthodes (ou fonctions) représentant respectivement les aspects déclaratifs et procéduraux de cette classe. Ce sont les méthodes qui assurent l'implantation des différentes facilités accessibles à l'usager par l'intermédiaire des menus. Chaque objet possédant à la fois les aspects déclaratifs et procéduraux de sa classe, il devient dès lors possible d'inspecter et de modifier le contenu d'un objet sur une base individuelle. A la figure 2.5, on présente par exemple une fenêtre d'inspection pour un noeud et une fenêtre d'inspection pour un arc. On voit donc qu'il est possible en tout temps de connaître les principales caractéristiques des objets impliqués dans la définition d'un problème.

NOEUD 6	
NUMERO	6
IDENTIFICATEUR	N0016
COORDONNEES	(149 428)
ACCES	NIL
ORIGINE	8
VEHICULE	V2
DEMANDE	100
TDEM	0
TOPT	100.0
FENETRE-TEMPS	NIL
TEMPS	2.940375
DIJKS-DISTMIN	0

ARC (6 2)	
NUMERO	4
IDENTIFICATEUR	A0024
NOEUDSOURCE	N0016
NOEUDCIBLE	N0012
LONGUEUR	150.65
ANGLE	264
RUE	NIL
INTERDICTION	NIL

Fig. 2.5 Fenêtres d'inspection pour un noeud et un arc

Nous allons maintenant préciser davantage le rôle de chacune des classes.

2.3.1 La classe RESEAU et ses classes d'appoint

a) RESEAU

Cette classe regroupe un ensemble d'attributs contenant les informations liées aux caractéristiques générales du réseau ainsi qu'un ensemble de méthodes assurant la gestion de ces informations. On y retrouve en particulier:

i) l'identificateur du réseau

ii) la liste des noeuds, des arcs et des rues

iii) la liste des arrêts, des origines et des noeuds simples

iv) la liste des véhicules desservant le réseau

v) un tableau permettant d'associer un numéro (ou étiquette) à chacun des noeuds

vi) une variable booléenne indiquant si des modifications ont été apportées à la structure du réseau

Cette information est importante car de telles modifications rendent désuètes les données touchant à la longueur des chemins plus courts entre les arrêts. L'usager doit donc savoir en tout temps si les informations qu'il possède reflètent ou non la situation courante.

vii) une variable booléenne indiquant si le réseau est euclidien

Cette information trouve son utilité lors de certaines manipulations interactives touchant à la structure du réseau. Ainsi, dans un réseau euclidien, le déplacement d'un noeud entraîne la mise-à-jour de la longueur des arcs adjacents à ce noeud.

Par ailleurs, les méthodes codées au sein de la classe RESEAU assurent une gestion appropriée des fonctions suivantes:

i) la transformation d'un noeud

L'usager est en mesure de modifier la classe à laquelle appartient un noeud particulier. Ainsi, un arrêt est susceptible de se transformer en noeud simple et il faut donc tenir à jour les listes de noeuds correspondantes.

ii) l'ajout ou le retrait de noeuds, d'arcs, de rues ou de véhicules

iii) la modification de certains attributs communs à l'ensemble des noeuds, arcs, rues ou véhicules

Cette facilité est utilisée afin d'uniformiser les caractéristiques de certains ensembles d'objets. C'est le cas par exemple si l'on veut que tous les véhicules aient une même capacité.

iv) la présentation d'informations générales concernant l'ensemble du réseau

Il est possible de faire apparaître dans la "fenêtre interactive" des informations générales telles: la liste des noeuds, arcs, rues ou véhicules, la liste des arrêts, l'ensemble des amas (arrêts desservis par un véhicule particulier), la liste des routes générées, etc.

v) la gestion du menu des rues et des véhicules

Ces menus sont gérés par le réseau et sont activés lorsque l'utilisateur sélectionne une fonction requérant la spécification d'une rue ou d'un véhicule particulier.

b) RESEAU-AFFICHAGE

La classe RESEAU-AFFICHAGE ajoute à la classe RESEAU toutes les propriétés essentielles à l'affichage du réseau et permet l'intégration des facilités offertes par le logiciel GRAPHER. On y retrouve en particulier la variable RESEAUGRAPHER dans laquelle est stockée la structure d'enregistrement utilisée par le logiciel pour la représentation graphique du réseau. GRAPHER permet également certaines modifications interactives à la structure du réseau telles:

i) l'ajout ou le retrait d'arcs

ii) l'ajout ou le retrait de noeuds

iii) le déplacement de la position des noeuds

La fenêtre du réseau est automatiquement mise-à-jour dès qu'une modification est apportée. De cette façon, toute transformation apparaît aussitôt à l'écran.

RESEAU-AFFICHAGE contient également les méhodes assurant:

i) l'initialisation appropriée de la fenêtre contenant le réseau

Cette fenêtre correspond à une occurrence de la classe FENETRE-RESEAU. Son titre, dans la bordure supérieure, correspond à l'identificateur du réseau affiché.

ii) l'affichage des rues du réseau

L'usager dispose d'un menu identifiant les différentes rues du réseau et il est possible de sélectionner une rue et d'en commander l'affichage. Cette rue apparaîtra alors en traits foncés à l'écran.

iii) l'affichage des parcours

Les parcours générés par les différents algorithmes sont affichés en traits foncés à l'écran. Cet affichage peut se faire selon deux modes distincts: le mode direct qui correspond à un trajet à "vol d'oiseau" et le mode exact qui emprunte les arcs du réseau. Dans ce dernier cas, le système doit évaluer, pour chacun des arrêts présents au sein des routes, le chemin plus court menant à l'arrêt suivant.

iv) l'affichage des amas (arrêts desservis par un véhicule)

Il est possible de visualiser rapidement l'ensemble des arrêts desservis par un véhicule particulier. Deux facilités sont disponibles à cet effet: le clignotement des arrêts à l'écran ou l'affichage d'un trait radial menant de l'origine à chacun des arrêts.

v) la modification des caractéristiques visuelles de l'ensemble des noeuds.

Les caractéristiques qui nous intéressent ici sont: la dimension en largeur et hauteur, la présence d'une étiquette numérique assurant l'identification du noeud et la fonte de l'étiquette. La coloration du noeud demeure inaccessible à l'usager car elle permet d'identifier la classe à laquelle appartient ce noeud. Ainsi, une origine est de couleur noire, un arrêt est blanc et un noeud simple est gris.

vi) la réduction des noeuds

L'usager a la possibilité de réduire les noeuds à l'état de simples points à l'écran. Cette possibilité est particulièrement intéressante pour les réseaux de grande dimension où les noeuds sont très rapprochés les uns des autres. Elle peut également s'avérer utile afin de réduire les noeuds appartenant à la classe NOEUDSIMPLE, ceux-ci ne présentant souvent aucun intérêt particulier.

vii) le déplacement du réseau à l'intérieur de la fenêtre d'affichage

Il est possible de produire un déplacement du réseau lorsque celui-ci excède les dimensions de sa fenêtre d'affichage. Ainsi, toute portion du réseau, affichée ou non, demeure accessible. Le déplacement peut se faire de deux manières différentes:

- un déplacement graduel horizontal ou vertical s'identifiant à un "scroll" classique. Si le réseau affiché est de grande dimension, ce type de déplacement risque toutefois de s'avérer particulièrement lent
- un déplacement orienté en fonction d'un noeud particulier. Ici, l'usager précise le numéro d'un noeud et le réseau se déplace de façon à ce que ce noeud occupe le centre de la fenêtre. Cette facilité permet un déplacement rapide d'une extrémité à l'autre du réseau

Mentionnons également la possibilité de positionner le réseau afin que les noeuds de coordonnées minimales en x et y occupent le coin inférieur gauche de la fenêtre d'affichage.

viii) la modification de l'échelle du réseau

Un "zoom" a été implanté à l'intérieur du système et permet d'augmenter ou réduire le coefficient d'agrandissement du réseau. La modification de l'échelle entraine un ajustement approprié de la longueur des arcs mais n'affecte pas la dimension des noeuds. Notons qu'une telle facilité permet de faire apparaitre les détails d'une zone particulière au sein d'un réseau de grande dimension

c) RESEAU-INTERFACE

Cette classe gère toutes les entrées et sorties du système. On y retrouve les méthodes autorisant:

i) la lecture et l'écriture des fichiers associés au réseau

Ces fichiers contiennent toutes les informations essentielles à la reconstitution du réseau, soit: un fichier décrivant la stucture du réseau, un fichier identifiant l'ensemble des arrêts et enfin un fichier contenant la table des distances.

Le fichier du réseau regroupe les informations touchant à l'ensemble des noeuds, des arcs et des rues. On y retrouve par exemple l'identificateur et les coordonnées de chacun des noeuds, la longueur et l'angle des arcs ainsi que la liste des arcs constituant chacune des rues. Lors de la lecture de ce fichier, un objet symbolique est créé pour chacun des noeuds, arcs ou rues présents et les informations associées permettent d'instancier cet objet. Lors de l'écriture, le processus est inversé et les informations contenues à l'intérieur des objets symboliques sont utilisées afin de produire le fichier désiré.

Le fichier des arrêts contient la description des points de demande à desservir (identificateur, demande, fenêtre de temps, etc.). Lors de la lecture, chaque noeud du réseau dont l'identificateur correspond à un identificateur d'arrêt se transforme, passant de la classe NOEUDSIMPLE à la classe ARRET. L'objet est par la suite instancié à l'aide des informations contenues dans le fichier.

Enfin, le fichier des distances contient la longueur des chemins plus courts entre les arrêts. Les informations pertinentes proviennent de la table des distances générée à l'aide de l'algorithme de Dijkstra.

Il faut remarquer qu'aucun des fichiers mentionnés ci-haut ne contient d'informations liées à l'application des algorithmes de génération de parcours. Plus précisément, aucune route n'est conservée et seul un fichier "objet" (décrit ci-après) permet de conserver ces routes.

ii) la lecture et l'écriture d'un fichier "objet"

Ce fichier assure la sauvegarde de tous les objets créés au sein de l'environnement. Comme ce sont ici les objets eux-mêmes qui sont conservés, cette facilité correspond de fait à la sauvegarde de l'environnement courant. Ainsi, toutes les informations y sont disponibles, y compris les routes produites à l'aide des algorithmes de génération de parcours. Il faut noter toutefois que les fichiers objet sont de dimension importante et leur chargement peut demander un temps considérable pour des réseaux de grande dimension.

iii) la gestion du fichier des résultats

Si l'usager le désire, les résultats obtenus suite à l'activation des algorithmes de génération de parcours peuvent être conservés dans un fichier spécial.

Par défaut, les résultats apparaissent au sein de la "fenêtre des résultats" Voici, à titre d'exemple, les informations présentées suite à l'activation d'un algorithme exact de commis-voyageur (ce résultat correspond à l'une des routes de la figure 2.1):

```
DISTANCE :  1020.81

ORDRE DE VISITE DES NOEUDS

    (8 3 6 2 1 12 5 4 9 10 16 8)

CHEMINEMENT EXACT

    ((8 3) (3 6) (6 2) (2 1) (1 12) (12 5) (5 4)
        (4 9) (9 10) (10 16) (16 9 8))
```

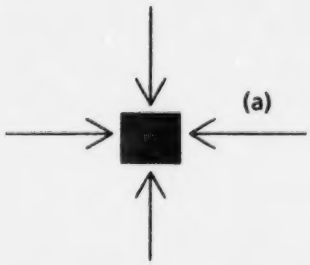
Fig. 2.6 Résultat d'un commis-voyageur

iv) la gestion de la fenêtre interactive

Si les informations demandées à l'usager sont succintes, le système utilise la fenêtre PROMPTWINDOW prédéfinie au sein de l'environnement INTERLISP-D. Par contre, pour des informations plus élaborées, une fenêtre spéciale est creée afin de faciliter l'interaction. Cette fenêtre est utilisée entre autres lors du processus d'instanciation de l'heuristique générale.

d) RESEAU-DIJKSTRA

La classe RESEAU-DIJKSTRA permet l'application de l'algorithme de Dijkstra lors de la recherche des chemins plus courts entre les arrêts. L'algorithme a eté implanté de telle façon qu'il autorise l'introduction de pénalités ou d'interdictions pour les virages a gauche et les virages en U. Il est également possible de préciser un accès sélectif à chacun des arrêts à l'aide d'un certain nombre d'arcs prédéterminés (figure 2.7). Suite à l'activation de cet algorithme, les résultats sont conservés dans chacun des noeuds concernés. Ainsi, chaque arrêt "connaît" la longueur des chemins plus courts menant à ses voisins.



Seul l'arc (a) permet d'atteindre l'arrêt

Fig. 2.7 Accès sélectif à un arrêt

L'activation de cet algorithme peut se faire selon deux modes:

i) mode local: l'algorithme est appliqué à l'ensemble des arrêts visités par un véhicule particulier.

ii) mode global: l'algorithme est appliqué à l'ensemble des arrêts définis au sein du réseau.

Les résultats sont conservés tant et aussi longtemps que l'usager ne commande pas une nouvelle application de cet algorithme. Il est donc possible que certains résultats deviennent périmés suite à des modifications telles:

i) l'ajout ou le retrait de noeuds et d'arcs

ii) le déplacement de la position des noeuds

iii) la modification de l'accès sélectif aux arrêts

iv) la modification de la longueur des arcs

v) la modification des pénalités de virage

vi) l'ajout ou le retrait d'interdictions de virage

C'est donc l'usager qui contrôle la mise-à-jour des informations concernant les chemins plus courts. Si aucune mise-à-jour n'est demandée, les algorithmes de génération de parcours utilisent les informations périmées afin de déterminer les séquences d'arrêts à visiter pour chacun des véhicules. Il est évident que ceci peut avoir une influence sur les séquences générées. Toutefois, le cheminement exact d'un véhicule est toujours évalué en tenant compte de toutes les modifications apportées à la structure du réseau.

Notons enfin que le système est en mesure de commander automatiquement l'exécution de l'algorithme de Dijkstra dans les situations suivantes:

i) L'usager demande l'activation d'un algorithme de génération de parcours alors que de nouveaux arrêts ont été introduits interactivement dans le réseau: le système commande alors la recherche des chemins plus courts pour chacun des arrêts qui ont été ajoutés.

ii) L'usager demande l'activation d'un algorithme de génération de parcours suite à la lecture d'un fichier d'arrêts (sans lecture concomitante d'un fichier des distances): le système commande alors l'activation de l'algorithme de Dijkstra en mode global car il ne dispose d'aucune table de distances appropriée.

e) RESEAU-COMVOY

Cette classe regroupe toutes les propriétés essentielles à l'application de l'algorithme exact de commis-voyageur. Il s'agit d'une implantation basée sur une approche de type "branch-and-bound" utilisant une technique de réduction de la matrice des distances [28]. Cet algorithme, relativement simple à implanter, est en mesure de résoudre en un temps raisonnable (10-20 minutes) des problèmes comprenant de quinze a vingt noeuds.

Les résultats obtenus sont conservés par le sytème et peuvent être reproduits tant et aussi longtemps que l'usager ne commande pas l'une des actions suivantes.

i) élimination d'arrêts ou de véhicules impliqués dans les parcours generés

ii) modification de l'affectation des arrêts aux véhicules

iii) modification de la structure du réseau

f) RESEAU-HEURISTIQUE

Cette classe assure l'implantation de l'heuristique générale et il en sera fait mention au chapitre 4.

2.3.2 La classe NOEUD, ses sous-classes et classes d'appoint

a) NOEUD

La classe NOEUD comprend un ensemble d'informations liées aux caractéristiques générales des noeuds du réseau, en l'occurrence: un identificateur, un numéro (utilisé comme étiquette lors de l'affichage) et la liste des arcs incidents. Cette classe regroupe de plus un ensemble de méthodes assurant la mise-à-jour des informations lors de certaines modifications apportées à un noeud, telles: la redéfinition de l'identificateur, l'ajout ou le retrait d'un arc incident, etc.

b) NOEUDSIMPLE

La classe NOEUDSIMPLE regroupe les noeuds que l'on identifie à l'intersection des rues dans un réseau urbain. Ces noeuds ne présentent habituellement aucun intérêt particulier et ne constituent que des points de transition entre les arrêts. Cette classe, relativement peu étoffée, contient les méthodes assurant l'inspection du contenu du noeud et sa transformation en arrêt ou origine.

c) NOEUDSPECIAL

La classe NOEUDSPECIAL permet de réunir les sous-classes ARRET et ORIGINE. Son utilité provient de ses classes d'appoint NOEUDSPECIAL-DIJKSTRA et NOEUDSPECIAL-COMVOY qui ne s'intéressent qu'aux noeuds dits "spéciaux" dans un réseau. Ces classes permettent de recueillir certains résultats obtenus suite à l'application de l'algorithme de Dijkstra et de l'algorithme exact de commis-voyageur. D'autres classes similaires pourraient bien sûr être introduites advenant l'intégration de nouveaux algorithmes d'optimisation.

d) ARRET

La classe ARRET regroupe les noeuds que l'on identifie aux points de demande desservis par la flotte de véhicules. La présence d'une demande et d'une fenêtre de temps pour le passage du véhicule sont les caractéristiques fondamentales qui distinguent les arrêts des autres types de noeuds. On y retrouve également le temps optimal pour le passage du véhicule, le temps alloué pour la cueillette de la demande et l'identificateur du véhicule desservant l'arrêt.

Les méthodes définies au sein de cette classe autorisent:

i) l'inspection du contenu de l'arrêt

ii) la transformation de l'arrêt en noeud simple ou origine

iii) la modification de la valeur des attributs

iv) l'affectation de l'arrêt à un véhicule particulier

L'usager peut, de manière interactive, associer un arrêt à l'un des véhicules disponibles. Comme un arrêt ne doit être desservi que par un seul véhicule, toute autre association se trouvera alors automatiquement rompue.

e) ORIGINE

Cette classe assure la description du noeud à l'intérieur duquel sont stockés les véhicules. Cette classe assure:

i) l'inspection du contenu de l'origine

ii) la transformation de l'origine en noeud simple ou en arrêt

iii) la gestion de la liste des véhicules

Notons que seuls les véhicules affectés à l'origine sont disponibles pour les algorithmes de génération de parcours. Cette affectation doit donc être faite de manière interactive par l'usager.

f) NOEUD-AFFICHAGE

La classe NOEUD-AFFICHAGE comprend toutes les méthodes permettant de modifier l'apparence d'un noeud à l'écran. Mentionnons à cet égard:

i) la modification de la dimension, en hauteur et largeur

ii) la reduction à l'état de simple point

iii) la modification de la fonte de l'étiquette

iv) le clignotement du noeud à l'ecran

Ces facilités sont accessibles au niveau de chaque noeud individuel. Il est donc possible de les utiliser afin de caractériser visuellement les noeuds auxquels on veut porter une attention particulière. A titre d'exemple, un arrêt de grande dimension peut s'identifier à un client important présentant une forte demande.

La classe NOEUD-AFFICHAGE assure également, pour chacun des noeuds, la gestion du menu des arcs incidents . Ce menu apparaît lorsque l'usager actionne le bouton centre de la souris à l'intérieur d'un noeud particulier. L'usager peut alors sélectionner un arc et appliquer à cet arc la fonction désirée.

g) NOEUDSPECIAL-DIJKSTRA

C'est à l'intérieur de cette classe que se trouvent les attributs et les méthodes permettant de conserver les résultats liés à l'application de l'algorithme de Dijkstra. Ainsi, chaque arrêt contient une table précisant la longueur des chemins plus courts menant à ses voisins. On y retrouve également certaines variables pour le stockage de résultats intermédiaires.

h) NOEUDSPECIAL-COMVOY

Cette classe a une fonction similaire à celle de NOEUDSPECIAL-DIJKSTRA pour l'algorithme exact de commis-voyageur. On y retrouve essentiellement un certain nombre de variables temporaires permettant de conserver des résultats intermédiaires.

2.3.3 La classe ARC

La classe ARC contient un ensemble d'attributs liés aux caractéristiques générales des arcs d'un réseau, en particulier: un identificateur, une longueur et un angle. On y retrouve également des méthodes assurant:

i) l'inspection du contenu de l'arc

ii) le calcul de la longueur et de l'angle d'un arc

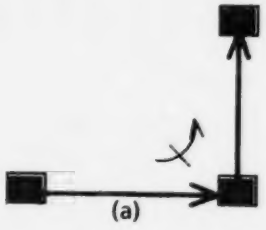
iii) la modification de la valeur des attributs

iv) l'intégration de l'arc au sein d'une rue

L'usager est en mesure de spécifier l'appartenance d'un arc à une rue particulière. Ce dernier sera alors ajouté à la liste des arcs définissant la rue (il en sera de même pour l'arc inverse, si celui-ci existe). Une telle facilité permet de définir interactivement l'ensemble des rues dans un réseau urbain.

v) la modification des interdictions de virages

Les interdictions pour les virages à gauche et les virages en U sont stockés au niveau des arcs et sont susceptibles d'être modifiées par l'usager (figure 2.8).



L'interdiction de virage à gauche
est stockée au niveau de l'arc (a).

Fig. 2.8 Interdiction de virage

2.3.4 La classe RUE

La classe RUE regroupe l'ensemble des rues présentes dans un réseau. Fondamentalement, une rue correspond à une liste d'arcs et les méthodes de cette classe assurent simplement la gestion appropriée de cette liste.

2.3.5 La classe VEHICULE

Cette classe regroupe l'ensemble des véhicules de transport desservant le réseau. Chaque véhicule possède les caractéristiques suivantes: une capacité, une vitesse moyenne et une autonomie maximale exprimée en termes de temps ou distance. De plus, on y retrouve une liste des arrêts à desservir.

Un véhicule est dit actif dès qu'il se trouve intégré au sein de l'origine. On peut alors générer des parcours pour ce véhicule à l'aide des différents algorithmes disponibles. Par ailleurs, un véhicule non intégré au sein de l'origine est dit inactif ou "en réserve" et ne peut être utilisé afin de desservir les arrêts.

La classe VEHICULE comprend les méthodes autorisant:

i) l'inspection du contenu du véhicule

ii) la modification de la valeur des attributs

iii) la gestion de la liste des arrêts desservis par le véhicule

iv) l'évaluation de la demande totale à desservir

v) l'activation ou la mise en réserve du véhicule

Notons en particulier que la mise en réserve d'un véhicule entraîne la "libération" de chacun des arrêts visités par celui-ci .

vi) le remplacement du véhicule

Il est possible de remplacer un véhicule actif par un autre véhicule, actif ou inactif. Le nouveau véhicule hérite alors de l'ensemble des arrêts et des routes associés à l'ancien véhicule. Lorsque les deux véhicules impliqués sont actifs, cette facilité a pour effet d'échanger les routes qui leur sont associées.

vii) la gestion des routes

Il s'agit ici d'éliminer de façon appropriée les routes devenues désuètes. Ceci se produit en particulier lorsque l'on modifie l'affectation des arrêts aux différents véhicules.

2.3.6 La classe ROUTE

On distingue ici deux types de routes: les routes produites par l'algorithme exact de commis-voyageur et les routes générées à l'aide de l'heuristique générale. La distinction est importante car les routes "heuristiques" doivent tenir compte des fenêtres de temps au niveau des arrêts afin d'évaluer correctement les temps de passage. Les routes de type "commis-voyageur" quant à elles ne se préoccupent aucunement de ces considérations d'ordre temporel, bien qu'éventuellement l'on puisse envisager l'implantation d'un algorithme de commis-voyageur respectant les fenêtres de temps.

Ajoutons que les routes n'existent qu'en fonction des véhicules chargés de les desservir. Ainsi, une route est automatiquement éliminée dès que le véhicule associé est mis en réserve ou détruit.

La classe ROUTE contient les méthodes permettant:

i) le calcul de la longueur de la route

ii) le calcul du temps de parcours

iii) l'évaluation du cheminement exact

En outre, il est possible de commander la création d'un fichier contenant toutes les informations associées à une route donnée, soit: la longueur de la route, le temps de parcours, les temps de passage aux arrêts et les caractéristiques du véhicule desservant cette route.

2.3.7 La classe FENETRE et ses sous-classes

La classe FENETRE autorise la spécialisation du système de fenêtres présents au sein de l'environnement INTERLISP-D. A titre d'exemple, ces fenêtres ont été définies de telle façon qu'il est impossible de les fermer à l'aide de la commande standard "Close". Ces fenêtres demeurent donc présentes à l'écran tant que l'usager ne demande pas spécifiquement de mettre fin à l'activation du programme ALTO. La classe FENETRE, comme nous l'avons déja mentionné, regroupe trois sous-classes spécialisées: FENETRE-RESEAU pour l'affichage du réseau, FENETRE-INTERACTIVE pour l'interaction avec l'usager et FENETRE-RESULTATS pour l'affichage des résultats obtenus suite à l'application des algorithmes de génération de tournées.

2.4 Les bénéfices liés à la programmation orientée objet

La representation sous forme d'objets symboliques des diverses entités impliquées dans la définition du problème se prête particulièrement bien au système que nous avons conçu. En effet, l'on traite ici des objets appartenant à des classes bien identifiées tels les noeuds du réseau, les arcs, les arrêts, les routes et les véhicules. Ainsi, chacun des objets est susceptible d'être modifié sur une base individuelle. C'est le cas par exemple lorsque l'usager désire modifier interactivement la valeur des attributs d'un véhicule ou d'un arrêt particulier. Comme toutes les informations pertinentes sont regroupées à l'intérieur d'un même objet, la mise-à-jour s'en trouve grandement facilitée.

De façon plus précise, la programmation orientée objet apporte les bénéfices suivants:

a) La possibilité d'intégrer à l'intérieur des objets autant d'attributs qu'on le désire. Ainsi, un véhicule se distingue par un certain nombre de caractéristiques telles la capacité, la vitesse ou la distance maximale autorisée. Cette liste n'est pas exhaustive et peut facilement être augmentée au besoin. Les attributs qui ne s'appliquent pas dans un contexte donné sont tout simplement ignorés, en

ce sens que leurs valeurs n'interviennent pas lors de la génération des tournées.

b) La spécialisation des classes d'objets s'avère ici fort utile. Elle permet en particulier de partitionner l'ensemble des noeuds en arrêts, origines et noeuds simples. Cette spécialisation des noeuds dans un réseau de transport constitue un concept fondamental pour lequel la programmation orientée objet est particulièrement bien adaptée. Le même principe pourrait d'ailleurs être utilisé afin de spécialiser la classe des véhicules en sous-classes dotées de caractéristiques spécifiques (camion, automobile, etc...)

   Notons que cette spécialisation assure une partition logique des différentes fonctions. Ainsi, les méthodes qui s'appliquent indistinctement à tous les noeuds, telles la modification de l'identificateur ou des coordonnées, sont stockées au sein de la classe générale NOEUD. Les méthodes qui ne s'appliquent qu'à un type spécifique de noeuds sont, pour leur part, stockées dans la sous-classe correspondante. Ainsi, la méthode qui assure la modification de la demande se trouve dans la classe ARRET et les autres noeuds n'y ont pas accès. Dès lors, il devient possible de concevoir des menus reflétant les diverses facilités présentes dans chacune des classes.

c) Par le biais des fenêtres d'inspection, l'usager est en mesure d'examiner le contenu des objets qui l'intéressent, ce qui ajoute au système un aspect de type "banque de données répartie". Cette facilité est facile à implanter car les objets symboliques constituent en eux-mêmes de petites entités individuelles regroupant un certain nombre d'attributs.

d) L'utilisation des valeurs de defaut est mise à profit afin de caractériser des occurrences typiques d'objets. Ainsi, certaines valeurs "standards" sont associées aux attributs d'un véhicule ce qui facilite la création d'une flotte homogène. Evidemment, l'usager est en mesure de modifier ces valeurs, s'il le désire.

e) Les classes d'appoint, qui ont pour but d'ajouter à une classe certaines propriétés supplémentaires, assurent le regroupement des fonctions apparentées. Il s'agit ici d'un outil de structuration non négligeable pour des programmes de dimension importante.

f) L'extension à d'autres domaines utilisant des réseaux se trouve facilitée. Notre système peut ainsi répondre à des applications dans le domaine des réseaux téléinformatiques. Il suffit pour cela de spécialiser la classe NOEUD en fonction des différents types d'ordinateurs présents (Hôte, IMP, etc...)

Ayant présenté la structure interne du système ALTO et le mode de représentation des éléments impliqués dans la définition d'un problème, nous allons maintenant nous intéresser à la description proprement dite de l'heuristique générale. Les concepts théoriques et pratiques à la base du développement de cette heuristique seront donc l'objet des deux chapitres suivants

Chapitre 3. L'heuristique générale : concepts théoriques.

Il existe actuellement de nombreux algorithmes conçus afin de résoudre les problèmes de génération de tournées pour un ou plusieurs véhicules [4]. Notre but est ici de caractériser les approches utilisées à l'aide d'un certain nombre de concepts fondamentaux s'intégrant au sein d'une heuristique générale. Cette heuristique s'identifie en quelque sorte à une structure de base que l'usager instancie avec les formules de son choix. Une telle approche paraît fort avantageuse pour deux raisons majeures :

a) Devant un problème particulier, l'approche de résolution susceptible de mener à une solution satisfaisante n'est pas toujours évidente. L'utilisation d'une structure générale que l'utilisateur instancie selon ses besoins et exigences apporte au système une souplesse que l'on ne retrouve pas dans les programmes usuels. Ceux-ci sont en effet "figés" dans une approche de résolution spécifique et toute modification de leur comportement implique nécessairement une modification du code (souvent au prix d'un effort considérable).

   L'heuristique générale, quant à elle, se prête très facilement à des expérimentations diverses par simple modification des formules d'instanciation. L'usager est ainsi en mesure de tester rapidement diverses stratégies de résolution de façon à identifier celles qui semblent les plus prometteuses.

b) Comme l'heuristique générale ne repose sur aucune approche de résolution spécifique, une instanciation adéquate assure la reproduction de nombreux algorithmes classiques décrits dans la littérature. En se basant sur la classification de Bodin & al. [4], mentionnons :

   i) les algorithmes basés sur les méthodes d'insertion [33]

      - nearest insertion
      - farthest insertion
      - cheapest insertion
      - greatest angle insertion
      - etc...

ii) l'algorithme du plus proche voisin ("nearest neighbor")

iii) la version séquentielle de l'algorithme de Clarke & Wright [18]

iv) les algorithmes de génération d'amas

- algorithme de balayage de Gillett & Miller [17]
- algorithme de Ferland & Rousseau pour le transport scolaire [11]

Tous les concepts utilisés par les algorithmes mentionnés ci-haut sont susceptibles d'être intégrés au sein de notre heuristique à l'aide des formules d'instanciation appropriées.

Comme nous l'avons mentionné au chapitre 2, la classe des problèmes traités par l'heuristique générale s'identifie à des problèmes de cueillette où une flotte de véhicules, homogène ou non homogène, est regroupée au sein d'une origine unique afin de desservir un ensemble d'arrêts. La demande à chacun des arrêts est déterministe et doit être recueillie en totalité par le véhicule desservant l'arrêt. Trois types de contraintes peuvent être introduits au niveau de la définition du problème: contraintes de capacité des véhicules, contraintes d'autonomie maximale des véhicules en terme de temps ou distance et fenêtres de temps au niveau des arrêts. Le but ultime est évidemment la minimisation des coûts d'opération, la définition de ces coûts étant implicitement contenue au sein des formules fournies par l'usager.

Dans les sections qui suivent, nous présenterons les concepts théoriques à la base de l'heuristique générale tout en identifiant de façon particulière les points de décision au niveau desquels l'usager peut intervenir. Le chapitre suivant insistera davantage sur des considérations touchant à l'implantation de l'heuristique, tout en présentant des exemples illustrant les concepts proposés.

3.1 Le modèle général

La figure 3.1 présente de façon schématique les éléments constituants du modèle proposé ainsi que les relations qu'ils établissent entre eux :

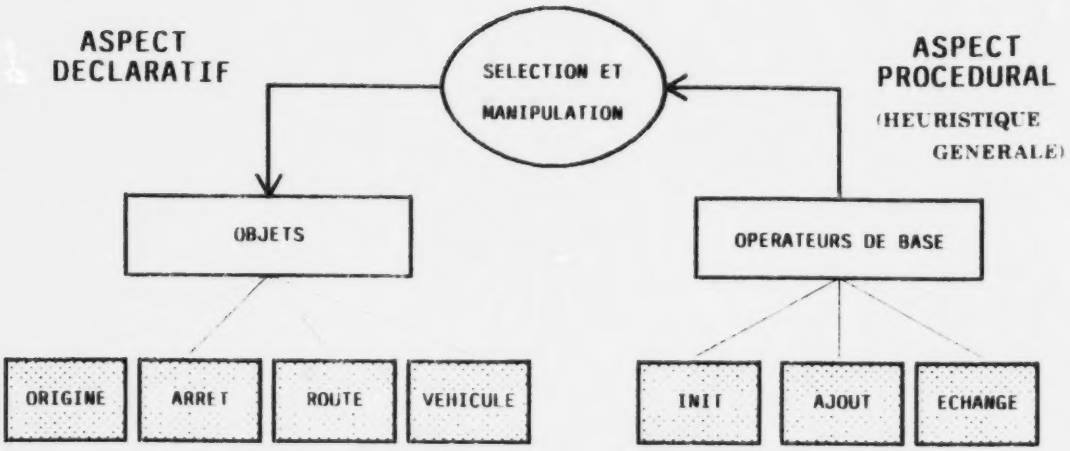


Fig. 3.1 Le modèle général

Plus précisément, nous retrouvons :

a) un ensemble d'objets

Les objets, auxquels sont associés des attributs, s'identifient à l'aspect déclaratif du modèle. Mentionnons :

i) l'origine et ses attributs :

- coordonnées
- nombre de vehicules disponibles

ii) les arrets et leurs attributs :

- coordonnées
- demande
- fenêtre de temps
- temps optimal (temps idéal pour l'arrivée du véhicule)
- durée du service (temps alloué afin de recueillir la demande)

iii) les routes et leurs attributs :

-longueur
-compacité (i.e. la somme des distances menant du centroïde à chacun des noeuds sur la route. Le centroïde s'identifie ici au noeud qui permet de minimiser cette somme [11])
-demande totale
-nombre d'arrêts

iv) les véhicules et leurs attributs :

- capacité
- vitesse
- distance maximale
- temps de parcours maximal

b) les opérateurs de base

De façon classique, l'exécution de l'heuristique générale se divise en trois phases principales :

i) phase d'initialisation des routes
ii) phase de construction des routes
iii) phase d'échange

Chacune de ces phases correspond conceptuellement à un opérateur de base qui sélectionne et manipule les objets appropriés. Plus précisément, nous retrouvons les opérateurs suivants :

i) INIT : opérateur assurant la sélection des germes ("seeds") nécessaires à l'initialisation des routes.

ii) AJOUT : opérateur assurant l'insertion de nouveaux arrêts à l'intérieur des routes.

iii) ECHANGE : opérateur permettant de modifier la séquence des arrêts visités à l'intérieur des routes.

Les opérateurs de base, qui s'identifient à l'aspect procédural du modèle, constituent évidemment le noyau de l'heuristique générale et seront analysés de façon détaillée dans les sections 3.2, 3.3 et 3.4.

c) un mécanisme de sélection des objets

L'intervention de l'usager se situe prioritairement au niveau de la sélection des objets appropriés. Pour ce faire, l'usager spécifie des formules qui associent une valeur à certains objets impliqués dans la définition du problème. Cette valeur correspond de fait à un coefficient de mesure de l'objet (CMO) et il devient alors possible d'ordonnancer les objets et de procéder à une sélection, en retenant par exemple l'objet possédant le CMO maximal ou le CMO minimal.

Il est important de noter que les formules définies par l'usager sont intégrées au sein des opérateurs de base précédemment mentionnés. De ce fait, toute modification apportée à ces formules ne peut qu'influencer fortement le comportement des opérateurs associés et entrainer l'heuristique vers de nouvelles avenues de résolution.

Les formules les plus couramment utilisées s'identifient à des fonctions numériques appartenant à l'un des types suivants :

i) $f : E_n \rightarrow \mathbb{R}$ (p.e. la demande à un noeud)

ii) $f : E_r \rightarrow \mathbb{R}$ (p.e. la longueur d'une route)

iii) $f : E_v \rightarrow \mathbb{R}$ (p.e. la capacité d'un véhicule)

iv) $f : E_n \times E_n \rightarrow \mathbb{R}$ (p.e. la distance entre deux noeuds)

v) $f : E_n \times E_r \rightarrow \mathbb{R}$ (p.e. la distance separant un noeud d'une route)

où

E_n = ensemble des noeuds (origine et arrêts)

E_r = ensemble des routes

E_v = ensemble des véhicules

Ayant présenté les concepts à la base du modèle général, il nous reste maintenant à préciser davantage le fonctionnement des opérateurs de base et identifier les points de décision où l'usager peut intervenir.

3.2 L'opérateur d'initialisation

L'opérateur INIT permet d'identifier un ou plusieurs germes assurant la création et l'initialisation d'une nouvelle route . Les germes sont choisis au sein de l'ensemble des arrêts "libres", soit les arrêts qui ne sont affectés à aucun véhicule (dans la suite, l'ensemble des arrêts libres sera représenté par le symbole E_k et tout sous-ensemble de E_k par SE_k).

La figure 3.2 illustre le principe d'initialisation: parmi l'ensemble des arrêts libres, représentés à l'aide de carrés blancs, n_2 et n_4 permettent de générer la route R_x tandis que n_5 et n_7 assurent respectivement la génération des routes R_y et R_z. Comme on le voit, les germes permettent de définir une orientation générale pour chacune des nouvelles routes créées.

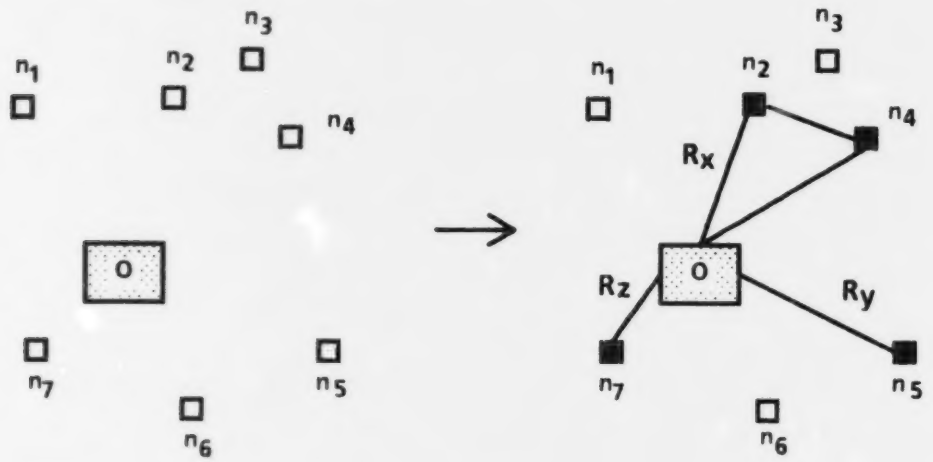


Fig. 3.2 Initialisation des routes R_x, R_y et R_z

La figure 3.3 présente une description détaillée de l'opérateur INIT.

INIT

N ← **nombre de routes**;

SE_k ← **sous-ensemble des arrêts libres**;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← **sélection des germes**;

v ← **sélection du véhicule**;

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N - 1;

Fig. 3.3 Opérateur INIT

3.2.1 Les points de décision

L'intervention de l'usager se situe au niveau des portions de l'algorithme présentées en traits foncés : il est donc possible de spécifier le nombre de routes désirées, le sous-ensemble des arrêts libres qui seront candidats lors du processus de sélection des germes ainsi que les formules assurant la sélection du véhicule et des germes pour chacune des nouvelles routes créées. De façon plus précise, nous retrouvons:

a) la spécification du nombre de routes

A ce niveau, l'usager spécifie le nombre de routes qu'il désire créer. L'opérateur d'initialisation retiendra le minimum entre cette valeur et le nombre de véhicules disponibles.

b) la spécification d'un sous-ensemble des arrêts libres (SE_k)

Il est possible de restreindre l'ensemble des arrêts libres à un sous-ensemble obéissant à certaines conditions définies par l'usager. Le processus de sélection des germes ne portera alors que sur les arrêts présents au sein de ce sous-ensemble. Cette formule est intéressante si l'on veut traiter prioritairement certains arrêts bien identifiés, tels les arrêts présentant la plus forte demande ou dont la fenêtre de temps est la plus étroite. En

l'absence de spécification, c'est évidemment l'ensemble de tous les arrêts libres qui sera retenu.

c) la sélection des germes

Comme on l'a vu à la figure 3.3, l'initialisation d'une route se fait à l'aide d'un ou plusieurs germes choisis au sein de l'ensemble SE_k, tel que défini en b). La sélection des germes permet de préciser l'orientation générale des routes qui seront construites lors de l'application subséquente de l'opérateur AJOUT.

d) la sélection du véhicule

L'usager est en mesure d'identifier le véhicule chargé de desservir la nouvelle route créée. Si l'on dispose d'une flotte hétérogène, il devient possible de retenir un véhicule en tenant compte des germes qui ont été précédemment sélectionnés (on retiendra par exemple le véhicule de capacité maximale si les germes affichent une forte demande). En l'absence de toute spécification, un choix aléatoire sera fait parmi l'ensemble des véhicules disponibles : une telle approche trouve son utilité dans le cas d'une flotte homogène où tous les véhicules sont équivalents.

Notons enfin que le processus d'initialisation peut être simulé manuellement par l'usager. A l'aide des menus, ce dernier peut en effet associer interactivement un ou plusieurs arrêts à chacun des véhicules disponibles. Cette alternative s'avère intéressante puisqu'un utilisateur expert est souvent en mesure d'initialiser les routes d'une façon beaucoup plus judicieuse que toute autre approche de nature algorithmique [13].

3.3 L'opérateur d'ajout

L'opérateur AJOUT correspond fondamentalement à un mécanisme itératif de construction des routes. A chaque itération, deux décisions fondamentales doivent être prises: il faut d'abord sélectionner un arrêt libre puis inserer cet arrêt à l'endroit désiré. Evidemment, c'est l'usager qui oriente l'opérateur d'ajout à l'aide des formules d'instanciation appropriées. La figure 3.4 fournit un exemple de ce principe pour les routes présentées à la figure 3.2. Ici, l'arrêt n_3 est retenu et inséré entre les arrêts n_2 et n_4 au sein de la route R_x. De façon classique, la procédure sera ainsi répétée jusqu'à épuisement de l'ensemble des arrêts libres.

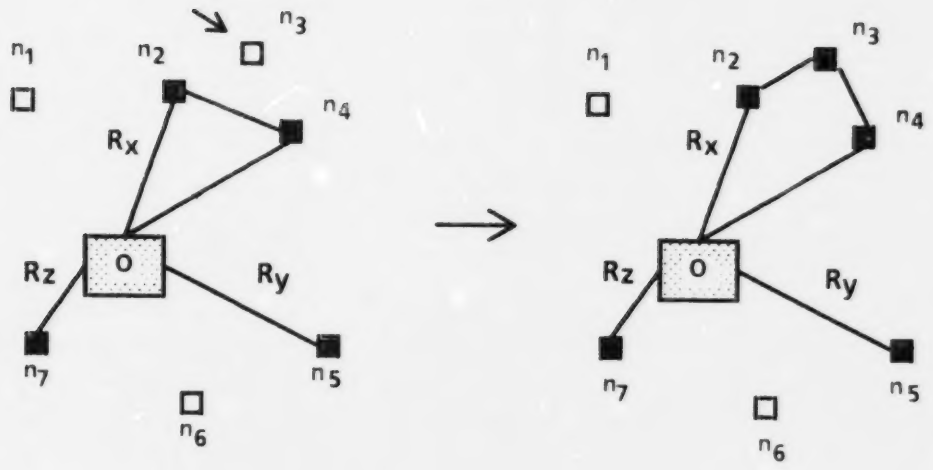


Fig. 3.4 Sélection et insertion de l'arrêt n_3

La figure 3.5 présente une description détaillée de l'opérateur AJOUT.

AJOUT

SE_k ← **sous-ensemble des arrêts libres**,
Tant que $SE_k \neq \emptyset$ faire
    r_1 ← **sélection de route 1**;
    k ← **sélection d'un arrêt libre**,
    r_2 ← **sélection de route 2**,
    i ← **sélection d'un arrêt sur une route**,
    si **critère d'arrêt** vérifié alors sortir de la boucle et terminer,
    si l'insertion de k respecte les contraintes alors
        insérer k **avant / après** i
    sinon
        traiter k selon **mode**;
    retirer k de SE_k;

Fig. 3.5 Opérateur AJOUT

Encore une fois, l'usager intervient au niveau des portions de l'algorithme en traits foncés.

3.3.1 Les points de décision

a) la spécification d'un sous-ensemble des arrêts libres (SE_k)

Tout comme pour l'opérateur INIT, l'ensemble des arrêts libres peut se réduire à un sous-ensemble satisfaisant certaines conditions définies par l'usager. L'opérateur d'ajout ne considérera alors que les arrêts présents dans ce sous-ensemble. Cette spécification est fort utile car elle permet de traiter les arrêts libres de façon sélective.

Il devient possible par exemple d'appliquer une instanciation $AJOUT_1$ de l'opérateur AJOUT à un sous-ensemble des arrêts libres situés au voisinage de l'origine et appliquer par la suite une nouvelle instanciation $AJOUT_2$ aux arrêts restants. L'usager est ainsi en mesure de modifier la stratégie de résolution en l'adaptant aux caractéristiques des arrêts traités.

b) la sélection des routes et des arrêts

La sélection des routes et des arrêts correspond à un point de décision majeur au sein de l'opérateur AJOUT. Comme nous l'avons mentionné précédemment, un mécanisme fondamental de sélection et insertion s'applique aux arrêts libres contenus dans l'ensemble SE_k. Cette procédure se répète ainsi jusqu'à épuisement de l'ensemble ou jusqu'à la vérification d'un critère d'arrêt défini par l'usager (voir le point d). A chacune des itérations, nous retrouvons :

i) la sélection d'une première route (r_1)

Il est possible d'identifier ici une route particulière, de façon à subordonner à cette route la sélection subséquente des arrêts k et i (tels que définis plus loin). Un usager peut, par exemple, sélectionner la route la moins longue, identifier l'arrêt libre situé le plus près de cette route et enfin procéder à l'insertion de l'arrêt à l'intérieur de la route. La spécification d'une route à ce niveau

permet donc de reproduire une séquence d'événements du type : sélection d'une route, sélection d'un arrêt libre et insertion de l'arrêt libre au sein de la route. Notons toutefois que la formule de sélection pour la route r_1 est optionnelle et se trouve inactive si sa valeur est fixée à NIL..

ii) la sélection d'un arrêt libre (k)

La formule de sélection définie par l'usager permet de retenir un arrêt particulier dans l'ensemble SE_k. L'arrêt libre sera par la suite traité de manière à l'inclure au sein d'une route. Cette formule est fondamentale car elle permet de définir l'ordre dans lequel les arrêts seront inclus au sein des routes.

iii) la sélection d'une seconde route (r_2)

Il est possible d'identifier ici une seconde route et de lui subordonner la sélection de l'arrêt i. Un usager peut, par exemple, sélectionner un arrêt libre, identifier la route la plus proche et insérer l'arrêt libre à l'intérieur de cette route. La spécification d'une route à ce niveau permet donc de reproduire une séquence d'événements du type : sélection d'un arrêt libre, sélection d'une route et insertion de l'arrêt libre au sein de la route. Notons toutefois que la formule de sélection pour r_2 est optionnelle et se trouve inactive si sa valeur est fixée à NIL.

iv) la sélection d'un arrêt sur une route (i)

La formule de sélection permet ici de retenir un arrêt particulier sur l'une des routes. L'identification d'un tel arrêt combinée à la spécification d'un ordre d'insertion définit de façon non équivoque un endroit pour l'insertion de l'arrêt libre k. Si une spécification de route est active, la sélection de l'arrêt i se fera à l'intérieur de cette route, sinon c'est l'ensemble de toutes les routes qui sera considéré.

c) la spécification d'un ordre d'insertion

Après avoir identifié un arrêt libre (k) et un arrêt sur une route (i), l'usager indique si k doit être inséré avant ou après i au sein de la route. Par défaut, k sera inséré après i.

d) la spécification d'un critère d'arrêt

Il s'agit en l'occurrence d'un prédicat qui retourne VRAI ou FAUX. Dès que le critère d'arrêt est vérifié l'opérateur AJOUT se termine, et ce, même s'il reste des arrêts à traiter. En l'absence de toute spécification, le critère d'arrêt est automatiquement non vérifié. Un tel critère peut se révéler utile si l'on désire interrompre le processus d'insertion des arrêts avant épuisement de l'ensemble SE_k (p.e. si la demande totale sur l'une des routes dépasse un seuil critique).

e) la spécification d'un mode de traitement de l'insertion

La spécification d'un tel mode est directement lié à la présence de contraintes et sera discuté en 3.3.2.

3.3.2 La spécification des contraintes

L'usager est en mesure d'activer trois types de contraintes : contraintes de capacité des véhicules, contraintes d'autonomie maximale des véhicules en terme de temps ou distance, fenêtres de temps au niveau des arrêts. A chacun de ces types correspond un paramètre de pondération permettant de contraindre davantage ou de relaxer la contrainte associée.

Dans la plupart des problèmes rencontrés en pratique, on retrouve souvent une ou plusieurs des contraintes mentionnées ci-haut. Leur degré d'élasticité est toutefois fort variable et l'on accepte parfois la violation d'une contrainte si la solution apporte par ailleurs de nombreux autres bénéfices. C'est là l'intérêt majeur des paramètres de pondération qui autorisent l'ajustement des contraintes en fonction du problème traité. Il faut noter que d'autres types de contraintes pourraient éventuellement être intégrées au sein du système sans qu'il y ait lieu de modifier les concepts à la base de l'heuristique générale. Nous pensons ici par exemple au transport de commodités diverses dans un véhicule compartimenté, à des restrictions portant sur les véhicules susceptibles de desservir un arrêt particulier, etc.

Actuellement, les cinq paramètres disponibles sont:

i) LAMBDAcap

pour chacune des routes,

DEMANDE TOTALE SUR ROUTE

≤ LAMBDAcap * CAPACITE DU VEHICULE

ii) LAMBDAdist

pour chacune des routes,

LONGUEUR DE LA ROUTE

≤ LAMBDAdist * DISTANCE MAXIMALE DU VEHICULE

iii) LAMBDAtemps

pour chacune des routes,

TEMPS DE PARCOURS

≤ LAMBDAtemps * TEMPS DE PARCOURS MAXIMAL DU VEHICULE

iv) LAMBDAfen1 , LAMBDAfen2

pour chacun des arrêts desservis par un véhicule,

a) cas symétrique : borne inférieure et supérieure

TEMPS D'ARRIVEE ≤ Topt + LAMBDAfen2 * (Tmax - Topt)

TEMPS D'ARRIVEE ≥ Topt - LAMBDAfen1 * (Topt - Tmin)

b) cas non symétrique : borne supérieure seulement.

TEMPS D'ARRIVEE ≤ Topt + LAMBDAfen2 * (Tmax - Topt)

c) cas non symétrique : borne inférieure seulement.

TEMPS D'ARRIVEE ≥ Topt - LAMBDAfen1 * (Topt - Tmin)

ou

Topt = temps optimal de service à l'arrêt

Tmin = borne inférieure de la fenêtre de temps

Tmax = borne supérieure de la fenêtre de temps

Comme on peut le constater, une valeur supérieure à 1 pour un paramètre donné correspond à une relaxation de la contrainte correspondante et une valeur inférieure à 1 produit un effet opposé. Par ailleurs, l'heuristique ignore les contraintes pour lesquelles la valeur du paramètre associé est fixée à NIL .

Evidemment, la présence de contraintes peut interdire l'insertion d'un arrêt libre à l'endroit spécifié. Dans un tel cas, divers modes de traitement de l'insertion s'offrent à l'utilisateur:

a) mode libre: l'arrêt ne sera pas inséré à l'endroit indiqué Ainsi, l'arrêt demeure libre et pourra éventuellement être traité au cours d'une phase ultérieure.

b) mode alternatif: l'heuristique utilise une formule alternative d'insertion (définie par l'usager) ne tenant compte que des endroits où l'insertion de l'arrêt respecte toutes les contraintes. S'il n'existe aucune alternative assurant le respect des contraintes, l'arrêt demeure libre.

c) mode insertion: l'arrêt libre est inséré à l'endroit spécifié malgré la violation des contraintes.

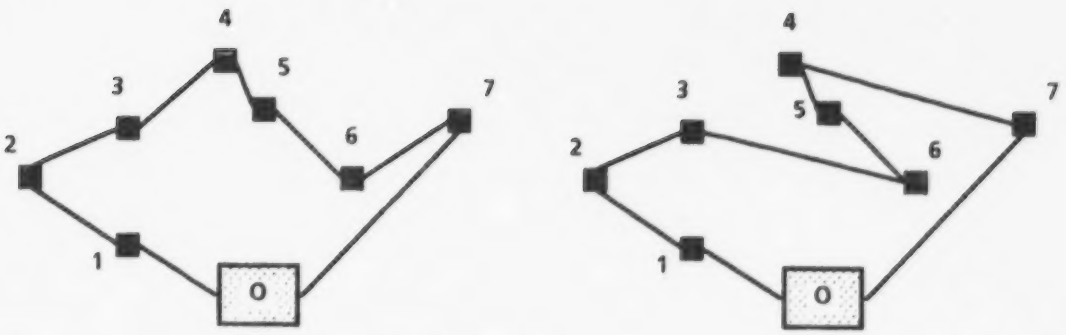
d) mode interruption: l'activation de l'opérateur AJOUT se termine dès qu'un arrêt ne peut être inséré à l'endroit indiqué. Ainsi, tous les arrêts non encore traités demeurent libres.

e) mode interactif: ici, l'usager doit spécifier l'un des quatre modes précédents dès que l'insertion d'un arrêt libre viole les contraintes. Cette approche interactive permet de traiter l'insertion de chacun des arrêts sur une base individuelle.

La présence de ces diverses modalités autorise un traitement plus raffiné de l'insertion pour certains problèmes présentant des contraintes multiples. Le mode "libre", par exemple, permet de relaxer les contraintes ou de modifier certaines formules afin de faciliter l'insertion des arrêts demeurés libres suite à une première activation de l'opérateur AJOUT.

3.4 L'opérateur d'échange

Cet opérateur assure la modification de la séquence des arrêts visités à l'intérieur des routes Il s'agit en l'occurrence d'un mécanisme d'échange basé sur une approche de type k-opt (figure 3.6).



échange des liens {(3,4) (6,7)} pour les liens {(3,6) (4,7)}

Fig. 3.6 Un échange de type 2-opt

La figure 3 7 présente de façon détaillée l'opérateur ECHANGE

ECHANGE

SEUIL ← $seuil_0$;

QUALITEmax ← 0;

SOLUTION COURANTE ← solution initiale;

Tant qu'il reste des échanges à considérer pour SOLUTION COURANTE faire

DIFF ← différence entre la longueur des liens ajoutés et retranchés

si DIFF ≤ SEUIL alors

SOLUTION ECHANGE ← solution obtenue après échange;

QUALITE ← évaluation de la **formule de qualité des routes** pour SOLUTION ECHANGE;

si QUALITE > QUALITEmax alors

SEUIL ← SEUIL - DIFF;

QUALITEmax ← QUALITE;

SOLUTION COURANTE ← SOLUTION ECHANGE;

Fig. 3.7 Opérateur ECHANGE

3.4.1 Les points de décision

a) la spécification d'un seuil

Comme on le voit à la figure 3.7, l'usager est en mesure de définir un seuil qui a pour fonction de restreindre l'ensemble des échanges considérés. Ainsi, pour une approche de type 2-opt, un échange ne sera pas retenu si la différence entre la somme des longueurs des deux liens ajoutés et la somme des longueurs des deux liens retranchés est supérieure au seuil. En fixant la valeur d'un tel seuil à U et en supposant que l'on dispose d'un réseau symétrique, la longueur totale des routes ne pourra augmenter de plus de U unités et ce, quels que soient les critères retenus afin d'évaluer la qualité des routes.

b) la formule de qualité des routes

L'usager intervient prioritairement au niveau de la spécification d'une formule de qualité pour les routes. Cette formule retourne toujours une valeur dans l'intervalle [0 , 1] et un échange est accepté dès qu'il permet de hausser cette valeur. Notons que les outils assurant la génération de la formule de qualité s'inspirent de techniques récentes en analyse de décision multicritères, techniques qui seront l'objet de la section 3.6.

3.4.2 Equivalence MTSP - TSP [3]

Il importe de retenir un mécanisme d'échange qui autorise à la fois des echanges intra-routes et inter-routes. La possibilité de transférer un ou plusieurs arrêts d'une route à une autre est fondamentale, puisque :

a) L'opérateur d'ajout introduit les arrêts à l'intérieur des routes en se basant sur des critères d'optimisation essentiellement locaux. Il faut donc un mécanisme qui permette de modifier cette affectation initiale, de façon à optimiser la solution dans une perspective globale.

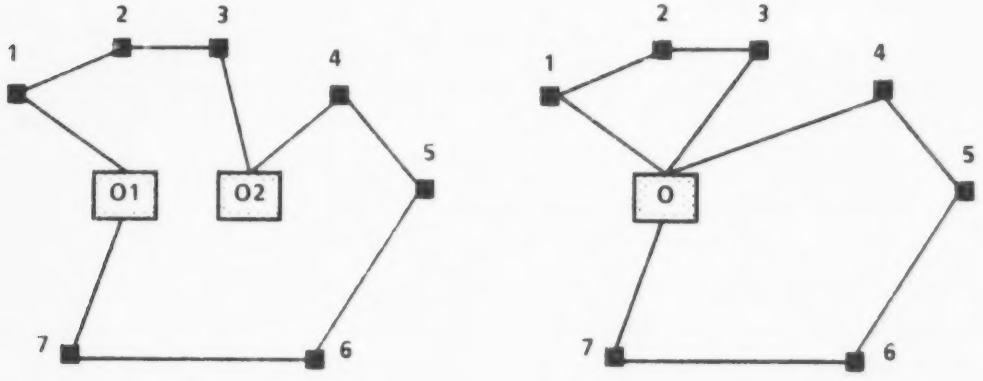
b) Certains des critères retenus pour la génération de la formule de qualité intéressent l'ensemble des routes (cf. section 3.6). C'est le cas par exemple si l'on cherche à équilibrer la demande desservie par les différents véhicules. Un

tel équilibre ne peut évidemment être atteint sans qu'il y ait un certain nombre d'échanges inter-routes.

A cet effet, nous avons retenu un mécanisme d'échange de type k-opt qui opère sur l'ensemble des routes générées. Cette approche s'appuie sur la modélisation d'un problème de type MTSP (Multiple Traveling Salesmen Problem) en un problème TSP (Traveling Salesman Problem) équivalent . Fondamentalement, la technique utilisée consiste à créer M copies de l'origine. Les liens unissant les noeuds du réseau à l'origine sont retirés et remplacés par M liens de longueur équivalente, soit un lien pour chacune des copies de l'origine. Toute solution au MTSP correspond alors à une route unique dans le réseau "augmenté" (figure 3.8a) et il devient possible d'appliquer une procédure d'échange de type k-opt sur cette route. Soulignons que les échanges considérés permettent alors le transfert de certains arrêts d'une route à une autre (figure 3.8b).

En pratique, notre système autorise l'application de procédures d'échange 2-opt et 3-opt sur chacune des routes individuelles ou sur la route unique présentée ci-haut. C'est donc l'usager qui décide de l'opportunité de procéder à des échanges inter-routes. A cet effet, il faut mentionner que la procédure classique k-opt telle que définie par Lin [27] a été étendue de façon à considérer un plus grand nombre d'échanges. L'analyse de cette approche nouvelle fera d'ailleurs l'objet de la section 3.5.

(a)



(b) Echange des liens {(1, 2) (7, O1)} pour les liens {(O1, 2) (7, 1)}

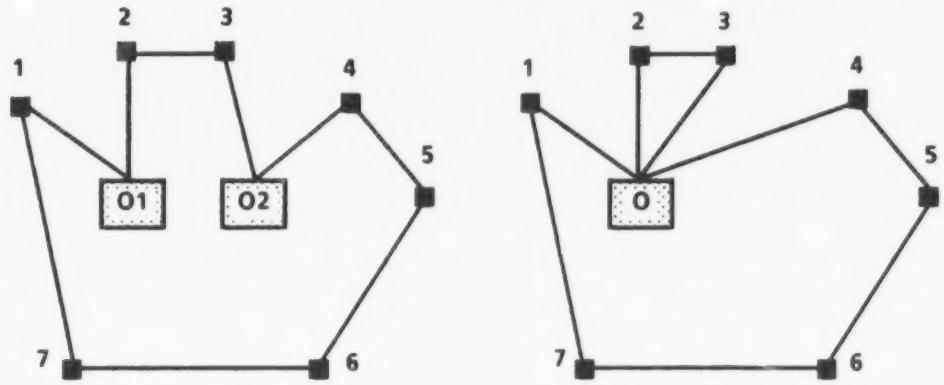


Fig. 3.8 Equivalence 2TSP - TSP

3.5 L'opérateur d'échange: une procédure k-opt étendue

La présente section introduit une technique généralisant l'approche de type k-opt pour des problèmes impliquant plusieurs véhicules. Tel qu'il sera expliqué plus loin, cette technique permet d'améliorer la performance générale de l'opérateur d'échange et s'avère même indispensable lorsque des fenêtres de temps sont présentes. Il ne s'agit toutefois que d'un raffinement technique qui ne touche en rien les concepts proposés précédemment.

Comme nous l'avons mentionné à la section 3.4, toute la conception de l'algorithme d'échange repose sur la modélisation d'un problème de type MTSP en un problème TSP équivalent. Il s'agit en l'occurrence de fusionner M routes en une "super-route" équivalente comprenant M copies de l'origine et sur laquelle peut s'appliquer une procédure d'échange k-opt. Nous cherchons ici à généraliser cette approche de façon à permettre la partition d'un tour unique en un certain nombre de sous-tours. Pour un problème de type TSP, ne comptant qu'une seule route, une telle partition mène nécessairement à une solution non réalisable Il n'en est pas de même toutefois pour un problème de type MTSP. De fait, toute partition devient ici réalisable s'il se trouve au moins une copie de l'origine sur chacun des sous-tours. A notre connaissance, cette observation n'a jamais été rapportée dans la littérature et l'analyse qui suit en démontre l'intérêt.

3.5.1 La procédure d'échange k-opt*

On voit à la figure 3.9 qu'il existe trois façons de relier deux sous-chaînes produites par le retrait de deux liens au sein d'un tour unique. Puisque la figure 3.9a reproduit le tour initial, la procédure 2-opt classique ne considère effectivement que le cas illustré à la figure 3.9b. De même, la figure 3.10 montre qu'il existe quinze façons de relier trois sous-chaînes et, de ce nombre, la procédure 3-opt classique ne considère que les cas 3.10b à 3.10h.

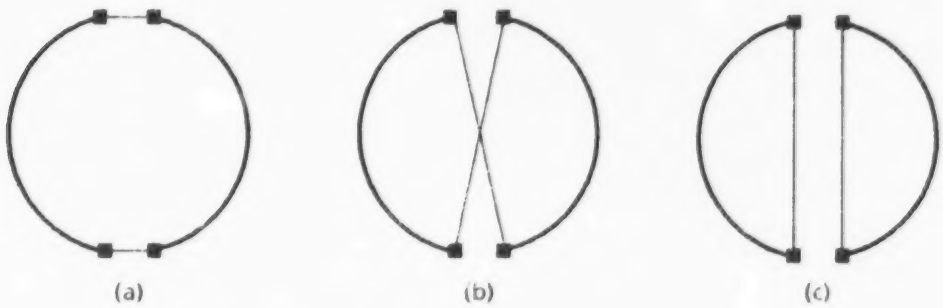


Fig. 3.9 Réunification de deux sous-chaînes

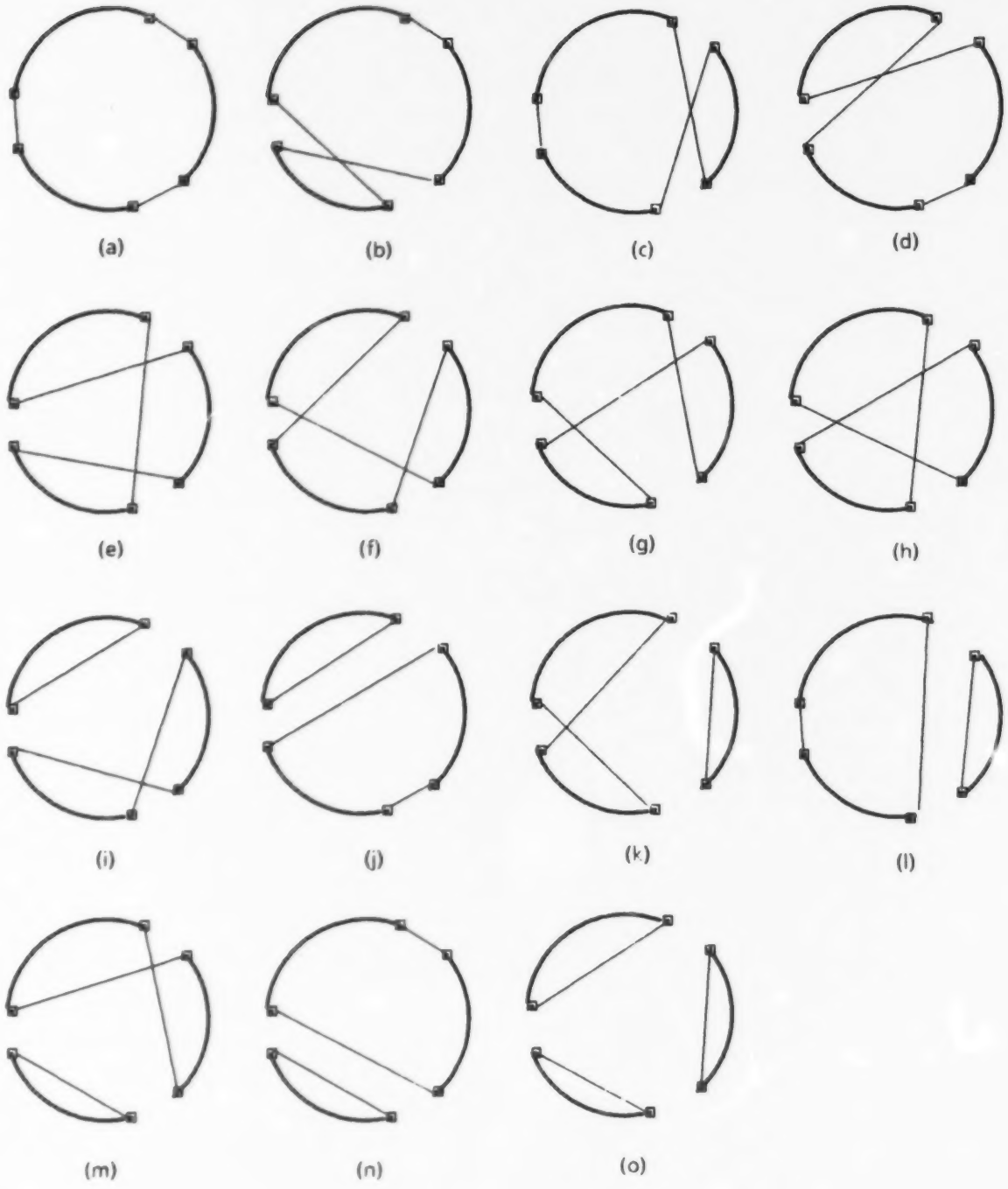


Fig. 3.10 Réunification de trois sous-chaînes

Pour les problèmes de type MTSP, certaines des connexions génératrices de tours multiples méritent une attention particulière. De fait, pour un échange de type 2-opt, nous pouvons observer les faits suivants:

a) si chacune des sous-chaînes contient au moins une copie de l'origine, alors tous les types de connexions décrits à la figure 3.9 sont réalisables.

b) si l'une des deux sous-chaînes ne contient aucune copie de l'origine, alors seuls les cas 3.9a et 3.9b sont réalisables.

De la même façon, pour un échange de type 3-opt:

a) si chacune des sous-chaînes contient au moins une copie de l'origine, alors tous les types de connexions décrits à la figure 3.10 sont réalisables.

b) si l'une des trois sous-chaînes ne contient aucune copie de l'origine, seuls douze types de connexions sont réalisables. Les types non autorisés entraînent la fermeture sur elle-même de la chaîne dépourvue d'origine (produisant ainsi un sous-tour dépourvu d'origine).

c) si deux sous-chaînes ne contiennent aucune copie de l'origine, alors seuls les cas 3.10a à 3.10h sont réalisables.

A la lumière de ces observations, il devient possible de considérer une procedure générale k-opt* tenant compte de tous les échanges menant à une solution réalisable, que celle-ci soit composée d'un tour unique ou de sous-tours multiples. Cette approche aura pour effet de doubler au maximum le nombre d'échanges considérés pour k = 2 et k = 3. La scission du tour ne pose ici aucun problème puisqu'à chaque solution comptant des sous-tours multiples correspond un tour unique équivalent que l'on obtient par un échange judicieux de liens jumeaux (deux liens sont dits "jumeaux" s'ils permettent de relier un même noeud à deux copies distinctes de l'origine, tel qu'illustré à la figure 3.11).

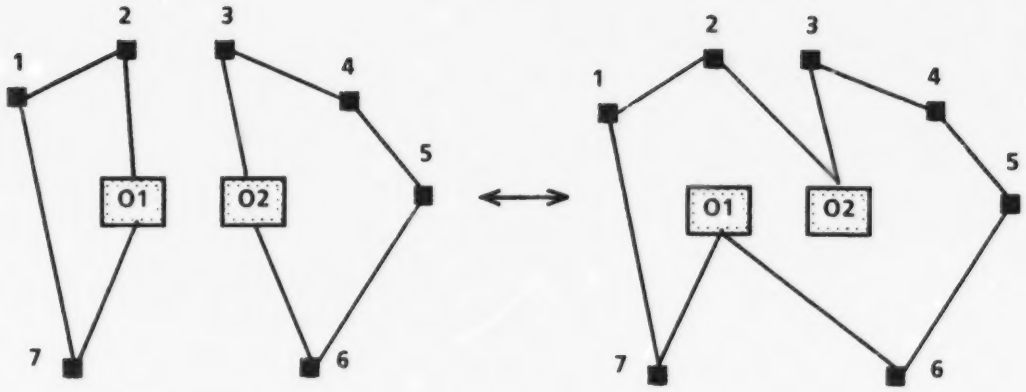
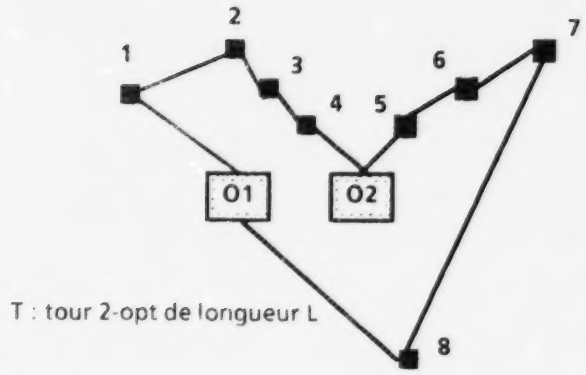


Fig. 3.11 Génération d'un tour unique

Utilisant la terminologie de Lin [27], un tour est dit k-opt* si sa longueur ne peut être réduite à l'aide de la procédure d'échange générale k-opt*. Il devient dès lors possible de conclure que:

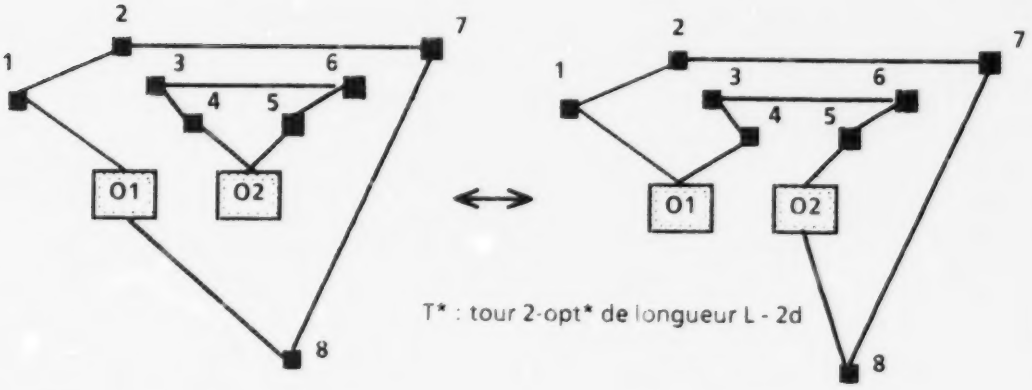
Tout tour k-opt* est k-opt

L'inverse n'est cependant pas vrai, comme le montre la figure 3.12.



T : tour 2-opt de longueur L

Echange des liens {(2,3) (6,7)} pour les liens {(2,7) (6,3)}



T* : tour 2-opt* de longueur L - 2d

$d(i,j) = x \qquad \forall (i,j) \in E$

$d(i,j) = x + 2d \qquad \forall (i,j) \notin E,\ (i,j) \neq (2,7), (6,3)$

$d(2,7) = d(6,3) = x - d$

où $E = \{\text{liens sur } T\} \cup \{(4,D1)\ (5,D1)\ (1,D2)\ (8,D2)\}$

$d > 0$

Fig. 3.12 tours 2-opt et 2 opt*

A la figure 3.12, quatre liens ont effectivement été échangés afin d'obtenir le tour 2-opt*, si l'on inclut ici la substitution de deux liens jumeaux assurant la réunification des deux sous-tours en un tour unique équivalent. Ainsi:

> **Un échange de type k-opt* peut mener à la substitution d'au plus 2k liens dans un tour défini sur le réseau "augmenté" (avec les copies de l'origine). En outre, si $k + x$ $(x > 0)$ substitutions ont lieu, au moins x d'entre elles impliquent des liens jumeaux.**

La procédure k-opt* est donc plus puissante que l'approche classique qui ne peut autoriser la substitution de plus de k liens dans un tour donné. De fait, il est facile d'observer que l'ensemble des échanges considéré par la procédure classique est strictement inclus au sein de l'ensemble considéré par la procédure générale.

3.5.2 Un algorithme itératif

La procédure d'échange k-opt* peut évidemment être incorporée au sein d'un algorithme itératif tel:

1) Générer une solution initiale
2) Réduire la longueur du tour à l'aide d'un échange de type k-opt*. Si l'échange génère plusieurs sous-tours, réunir ceux-ci de façon à former un tour unique équivalent.
3) Répéter l'étape 2 jusqu'à ce qu'il n'y ait plus d'amélioration possible.

La complexité de l'étape 2 de cet algorithme s'exprime de la façon suivante.

$$\binom{n}{k} (2^{k-1}(k-1/2)! - 1)$$

où n = nombre de liens dans le tour

$(k-1/2)! = (k-1/2) * (k-3/2) * \ldots * 3/2$

Pour un algorithme équivalent conçu afin de réaliser des échanges classiques de type k-opt, la complexité de l'étape 2 s'établit de la façon suivante:

$$\binom{n}{k} (2^{k-1}(k-1)! - 1)$$

Ainsi, pour une valeur de k fixée ($k > 1$), la complexité respective des procédures d'échange k-opt et k-opt* est de

$$C\binom{n}{k} \quad \text{et} \quad C'\binom{n}{k} \quad \text{où} \quad C < C'.$$

Il s'agit donc d'une complexité dans l'ordre de n^k pour les deux approches. Pour $k = 2$, nous obtenons par exemple un ordre de n^2 avec $C = 1$ et $C' = 2$. Ainsi, pour un nombre d'itérations équivalent, le temps d'exécution requis par l'algorithme général équivaudra approximativement au double du temps requis par l'algorithme classique et ce, indépendamment du nombre de liens présents au sein du tour. Cet aspect est fort intéressant puisque la complexité d'une procédure d'échange de type 3-opt est, comme on le sait, dans l'ordre de n^3.

3.5.3 Résultats expérimentaux

Afin de démontrer l'intérêt de la procédure k-opt*, nous avons mené quelques expériences à l'aide de réseaux générés aléatoirement et comprenant 20, 40, 60 et 80 noeuds (10 réseaux pour chaque dimension). Les tours initiaux ont été produits à l'aide d'une heuristique classique connue sous le nom de "farthest insertion" [33]. Par la suite, les algorithmes itératifs d'échange 2-opt, 2-opt* et 3-opt ont été appliquées successivement à chacun des tours initiaux afin de comparer leur performance respective. La figure 3.13a présente, pour chaque algorithme retenu et pour des problèmes 2TSP et 3TSP, la longueur des tours générés et le pourcentage d'amélioration par rapport à la solution initiale INIT. Chaque résultat correspond à une moyenne portant sur 10 problèmes distincts. Les temps d'exécution, exprimés en secondes, sont présentés à la figure 3.13b et l'on observe que si TEMPS(2-opt) = t et n = nombre de liens dans le tour alors TEMPS(2-opt*) ≈ 2t et TEMPS(3-opt) ≈ nt.

Les résultats de la figure 3.13 montrent la supériorité d'une approche de type 2-opt* par rapport à une approche de type 2-opt. Il faut toutefois noter que dans 60% des cas, les solutions obtenues furent identiques. Ceci signifie que la procédure d'échange 2-opt* ne garantit en rien l'obtention d'une solution de meilleure qualité. Il est toutefois relativement peu coûteux de l'utiliser et cette méthode est susceptible d'apporter des bénéfices appréciables.

De façon générale, il apparaît donc que la procédure 2-opt* représente un bon compromis entre les procédures classiques 2-opt et 3-opt pour ce qui est du rapport entre la qualité de la solution et le temps d'exécution.

2TSP	20	%	40	%	60	%	80	%
INIT	2253.65		2895.81		3257.47		3874.01	
2OPT	2194.64	2.6	2828.13	2.3	3185.81	2.2	3777.15	2.5
2OPT*	2159.38	4.2	2803.14	3.2	3156.49	3.1	3746.66	3.3
3OPT	2088.85	7.3	2742.81	5.2	3071.79	5.7	3641.44	6.0

3TSP	20	%	40	%	60	%	80	%
INIT	2500.17		3133.71		3488.88		4082.60	
2OPT	2338.31	6.5	2965.38	5.4	3338.86	4.3	3878.05	5.0
2OPT*	2283.96	8.6	2916.28	6.9	3286.53	5.8	3804.98	6.8
3OPT	2219.07	11.2	2837.22	9.5	3167.90	9.2	3643.00	10.8

Fig. 3.13 (a) longueur des tours

2TSP	20	40	60	80
2OPT	5.3	13.7	42.8	70.3
2OPT*	10.3	26.3	82.7	139.2
3OPT	117.5	595.6	2634.6	6130.1

3TSP	20	40	60	80
2OPT	8.1	19.5	46.0	88.2
2OPT*	17.2	39.7	96.2	192.4
3OPT	190.8	868.7	2971.6	7909.0

Fig. 3.13 (b) temps d'exécution (secondes)

3.5.4 L'ajout de contraintes

Nous avons precédemment introduit les procedures d'échange k-opt et k-opt* pour des problèmes de type MTSP ou aucune contrainte n'est specifiée. Il est toutefois possible de traiter à l'aide de cette approche des problèmes plus complexes comprenant des contraintes de capacité ou de distance maximale autorisée. Il suffit alors de verifier que ces contraintes sont

toujours respectées suite à un échange impliquant le transfert de certains arrêts d'une route à une autre.

Si l'on considère maintenant l'introduction de fenêtres de temps au niveau des arrêts, il faut observer que la procédure 2-opt classique se révèle alors de peu d'utilité lors d'échanges inter-routes. Tel qu'illustré à la figure 3.14a, l'approche 2-opt a pour effet de réunir les premiers arrêts de la route 1 (arrêts 1 et 2) aux premiers arrêts de la route 2 (arrêts 5 et 6). Puisque les arrêts situés en début de route sont en général ceux dont la borne supérieure de la fenêtre de temps est la moins élevée, il existe de fortes chances d'excéder cette borne aux arrêts 5 et 6, ces derniers étant maintenant visités après les arrêts 1 et 2. Une telle solution est donc fort probablement non réalisable. L'approche 2-opt* par contre se trouve mieux adaptée à un tel problème puisqu'elle considère les échanges permettant de réunir les premiers et les derniers arrêts de deux routes distinctes (figure 3.14b). Il est donc tout à fait probable que la nouvelle solution générée soit réalisable. Une telle observation ajoute évidemment beaucoup d'intérêt à l'approche générale pour des problèmes avec fenêtres de temps.

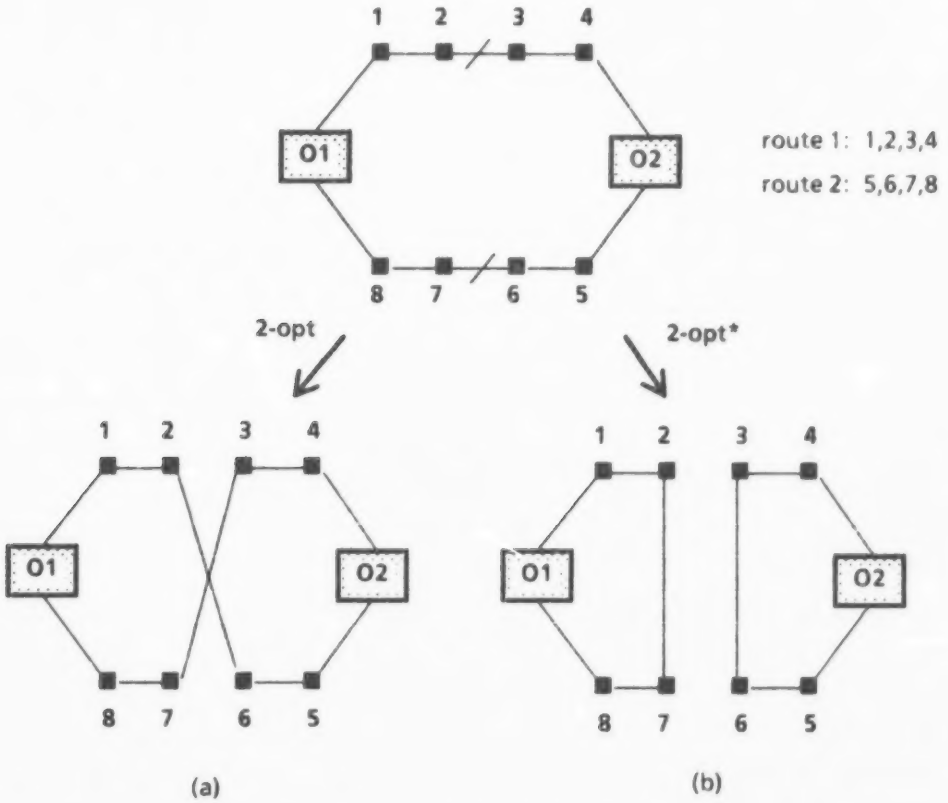


Fig 3.14 Echanges 2-opt et 2-opt*

3.6 L'opérateur d'échange: une approche multicritères [22]

L'opérateur d'échange tel que défini précédemment s'intègre dans une perspective d'analyse de décision multicritères. Les critères retenus lors de la génération de la formule de qualité pour les routes sont :

c1) la longueur

c2) la compacité (telle que définie en [11]).

c3) l'utilisation de la capacité du véhicule, i.e.

demande totale / capacité du véhicule

c4) l'utilisation de la distance maximale autorisée pour le véhicule, i.e.

longueur de la route / distance maximale du véhicule

c5) l'utilisation du temps de parcours maximal autorisé pour le véhicule, i.e.

temps de parcours / temps de parcours maximal du véhicule

c6) le temps d'arrivée du véhicule à chacun des arrêts

Fondamentalement, la technique utilisée afin de générer une formule de qualité consiste à agréger tous les critères en un critère unique modélisé sous la forme d'une fonction d'utilité globale. Pour ce faire, des fonctions d'utilité partielles sont construites pour chacun des critères retenus et la valeur retournée par la fonction globale équivaut à la somme des valeurs des fonctions d'utilité partielles.

Compte tenu du contexte particulier au sein duquel s'applique cette analyse multicritères, il nous a fallu modifier sensiblement les techniques usuelles. Celles-ci définissent en général deux valeurs extrêmes pour chaque critère de façon à borner la fonction d'utilité partielle. L'une de ces valeurs correspond à une valeur "idéale" pour le critère et retourne une utilité maximale. L'autre valeur a une interprétation opposée et, à ce niveau, l'utilité du critère est nulle (figure 3.15a).

Comme on peut le constater, toute valeur au-delà de la borne supérieure retourne une utilité maximale alors que toute valeur en-deçà de la borne inférieure retourne une utilité minimale. Dans notre cas, il est souvent impossible d'identifier deux valeurs extrêmes au-delà desquelles toute solution demeure ou "très bonne" ou "très mauvaise". Si l'on considère par exemple l'utilisation de la capacité des véhicules, l'on cherchera à éviter autant la surutilisation que

la sous-utilisation. De ce fait, il nous faut retenir une fonction d'utilité partielle qui s'apparente à celle présentée à la figure 3.15b.

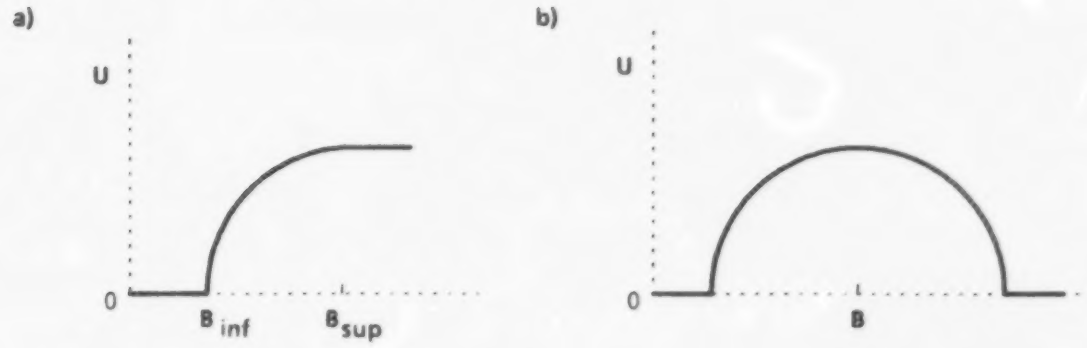


Fig 3.15 Fonctions d'utilité partielle (U) pour un critère de qualité

Ainsi, les paramètres utilisés lors de l'évaluation d'une fonction d'utilité partielle pour un critère donne c_i sont :

a) le poids du critère ($POIDS_{c_i}$)

Ce poids non négatif établit l'importance du critère en rapport avec les autres. Evidemment, un critère est ignoré si son poids est nul. De façon plus précise, on établit l'importance du critère c_i de la façon suivante :

$$IMP_{c_i} = \frac{POIDS_{c_i}}{\sum_{j=1}^{m} POIDS_{c_j}}$$

où m = nombre de critères

b) l'optimum [$BINF_{c_i}$, $BSUP_{c_i}$]

Cet optimum correspond à un intervalle à l'intérieur duquel toute valeur est jugée idéale pour le critère considéré. Si l'on désire que cet optimum s'identifie à une valeur unique, il suffit de poser: $BINF_{c_i} = BSUP_{c_i}$.

c) l'écart maximal toléré par rapport à l'optimum ($ECARTmax_{C_i}$)

Ce paramètre permet de borner la fonction d'utilité partielle. C'est en effet l'écart par rapport à l'optimum qui est utilisé afin d'évaluer l'utilité du critère. Ainsi, un écart équivalent à 0 retourne une utilité maximale alors qu'un écart supérieur à l'écart maximal retourne une utilité mimimale.

d) des multiplicateurs d'écart ($PENAL1_{C_i}$ et $PENAL2_{C_i}$)

Si la valeur du critere c_i appliqué à un objet o est moindre que la borne inférieure de l'optimum, on retient l'écart suivant:

$$ECART_{C_iO} = PENAL1_{C_i} * (BINF_{C_i} - VALEUR_{C_iO})$$

Si la valeur du critère c_i appliqué à un objet o est plus grande que la borne supérieure de l'optimum, on retient:

$$ECART_{C_iO} = PENAL2_{C_i} * (VALEUR_{C_iO} - BSUP_{C_i})$$

Les multiplicateurs permettent ainsi de penaliser davantage tout écart par rapport à l'optimum et de réduire en proportion l'utilité du critère. Ils peuvent s'avérer utiles si un écart présente une signification différente selon que l'on est au-dessus ou en-dessous de l'optimum. Par défaut, la valeur des multiplicateurs est fixée à 1.

e) ..t ensemble d'objets (E_{C_i})

Le critère c_i est applique à chacun des objets presents au sein de l'ensemble E_{C_i}. Pour les six critères de base precedemment mentionnés, nous avons:

i) longueur et compacité: ensemble des routes

ii) utilisation de la capacité, du temps de parcours maximal ou de la distance maximale: ensemble des véhicules

iii) temps d'arrivée: ensemble des arrêts

A l'aide de ces paramètres, il devient possible de créer une fonction d'utilité partielle f_{c_i}. Cette fonction, qui retourne l'utilité du critère c_i, se définit ainsi:

$$f_{c_i}(x_{c_i}) = \begin{cases} IMP_{c_i} * (1 - x_{c_i}^2) & \text{si } x_{c_i} \in [0, 1] \\ 0 & \text{sinon} \end{cases}$$

où

$$x_{c_i} = \frac{1}{|E_{c_i}|} \sum_{e \in E_{c_i}} \frac{ECART_{c_i e}}{ECARTmax_{c_i}}$$

$$ECART_{c_i e} = \begin{cases} \text{si } VALEUR_{c_i e} < BINF_{c_i} \text{ alors} \\ \quad PENAL1_{c_i} * (BINF_{c_i} - VALEUR_{c_i e}) \\ \text{si } VALEUR_{c_i e} > BSUP_{c_i} \text{ alors} \\ \quad PENAL2_{c_i} * (VALEUR_{c_i e} - BSUP_{c_i}) \\ \text{sinon } 0 \end{cases}$$

Comme on le voit, l'utilité du critère c_i, qui correspond à la valeur retournée par la fonction d'utilité partielle f_{c_i}, se trouve toujours dans l'intervalle [0 , IMP_{c_i}]. La fonction d'utilité globale f étant additive, nous avons :

$$0 \leq f(x_{c_1}, \ldots, x_{c_m}) = \sum_{i=1}^{m} f_{c_i}(x_{c_i}) \leq \sum_{i=1}^{m} IMP_{c_i} = 1$$

où m = nombre de critères

La formule de qualité des routes dont il a été fait mention n'est autre que cette fonction d'utilité globale définie à l'aide des six critères précédemment mentionnés et appliquée à la solution courante. Cette formule retourne donc toujours une valeur comprise entre 0 et 1 et la valeur croît lorsque la qualité augmente.

3.7 L'activation des opérateurs

Lorsque nous avons présenté l'heuristique générale à la section 3.1, nous avons identifié trois phases principales soit l'initialisation des routes, la construction des routes et la phase d'échange. Il est bon de noter que les opérateurs assurant la réalisation de ces différentes phases sont indépendants les uns des autres. De ce fait, ils ne sont soumis à aucune séquence d'activation prédéfinie. Lors de la résolution d'un problème, l'usager est ainsi en mesure d'appliquer les divers opérateurs de base dans l'ordre qui lui convient, la seule restriction provenant du fait que le premier opérateur de la séquence se doit d'être l'opérateur d'initialisation (figure 3.16).

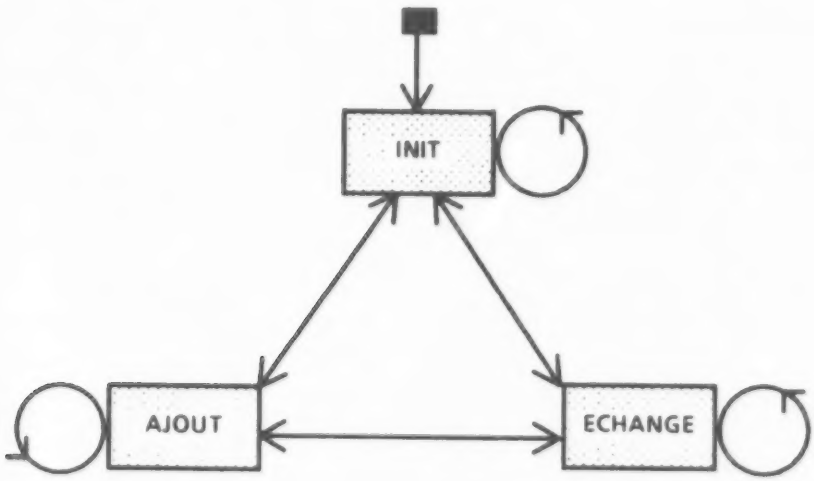
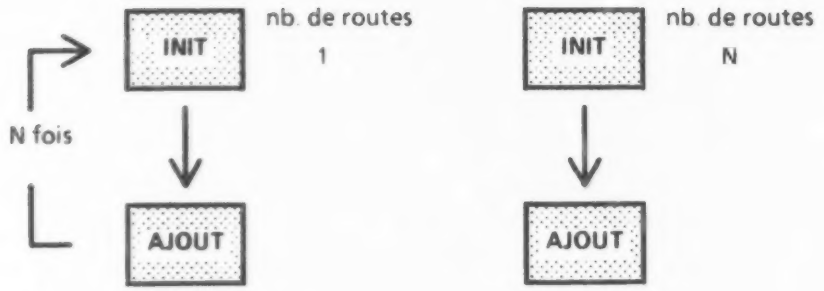


Fig. 3.16 Sequences d'activation des operateurs

Ces possibilités sont intéressantes pour des problèmes présentant des contraintes multiples et pour lesquels l'insertion de tous les arrêts libres à l'aide d'une seule application des opérateurs INIT et AJOUT est impossible. L'usager est alors en mesure de recourir à diverses stratégies afin d'assurer le traitement des arrêts demeurés libres. Il peut ainsi commander à nouveau l'application de l'opérateur INIT afin de créer une ou plusieurs nouvelles routes ou encore réactiver directement l'opérateur AJOUT après avoir modifié certaines formules et relaxé les contraintes. En outre, l'indépendance des opérateurs assure le développement de stratégies de résolution basées sur la génération séquentielle ou parallèle de routes (figure 3.17).



(a) approche séquentielle (b) approche parallèle

Fig. 3.17 Génération de N routes

L'heuristique générale, dans sa conception actuelle, offre donc une grande flexibilité à l'usager confronté à un problème particulier. Le chapitre suivant nous permettra maintenant d'étudier concrètement le fonctionnement de cette heuristique.

Chapitre 4. L'heuristique générale : concepts pratiques.

Ce chapitre traite des aspects touchant à l'implantation de l'heuristique générale. Il y sera fait mention des facilités et outils offerts à l'usager lors de la résolution pratique d'un problème. De plus, un certain nombre d'exemples illustrant les possibilités de l'heuristique seront présentés.

4.1 La classe RESEAU-HEURISTIQUE

Tel que nous l'avons souligné au chapitre 2, c'est la classe d'appoint RESEAU-HEURISTIQUE qui assure l'implantation de l'heuristique générale. Cette classe regroupe en particulier toutes les formules et tous les paramètres utilisés par les opérateurs de base. On y retrouve également les méthodes assurant :

a) la mise-à-jour des formules et paramètres.

b) la spécification d'un temps supplémentaire de déplacement.

Lors de la génération des routes à l'aide de l'heuristique générale, il est possible d'ajouter un temps supplémentaire pour chaque déplacement du véhicule entre deux arrêts. Un tel ajout témoigne de la prudence de l'utilisateur qui cherche à introduire une certaine flexibilité dans les horaires produits.

c) l'activation des opérateurs de base INIT, AJOUT et ÉCHANGE.

d) l'activation de l'opérateur INIT en mode "manuel".

Comme nous l'avons mentionné au chapitre précédent, l'usager est en mesure d'initialiser les routes en affectant interactivement les arrêts aux différents véhicules.

e) l'activation de l'opérateur AJOUT en mode "démonstration".

Ce mode permet à l'usager de suivre le déroulement de l'algorithme d'insertion à l'écran. Plus précisément, à chacune des étapes, certaines facilités graphiques sont activées. Mentionnons :

i) lors de la sélection d'un arrêt libre : clignotement de l'arrêt à l'écran.

ii) lors de la sélection d'un arrêt sur une route : clignotement de l'arrêt à l'écran.

iii) lors de l'insertion : les routes affichées sont mises-à-jour de façon à tenir compte de l'insertion du nouvel arrêt.

f) le support du retour arrière

Au cours du processus de résolution, il est toujours possible d'invalider les effets du dernier opérateur activé (figure 4.1). Cette facilité est fort utile lorsqu'un opérateur incorrectement instancié est utilisé.

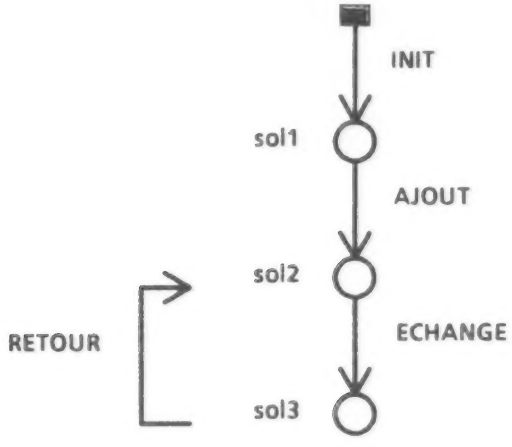


Fig. 4.1 Le retour arrière

g) le retour à une solution particulière

L'usager est en mesure de conserver les solutions intermédiaires obtenues au cours du processus de résolution. Chacune de ces solutions peut donc être réactivée au moment jugé opportun.

h) le support du gestionnaire des formules

S'il le désire, l'usager est en mesure de conserver les formules qu'il définit. Il suffit pour cela de leur associer un identificateur unique. A chaque identificateur correspond une entrée dans le répertoire des formules. Ce répertoire est supporté par un gestionnaire assurant les fonctions suivantes:

i) la création de nouvelles formules : pour chaque formule créée, une nouvelle entrée est introduite dans le répertoire

ii) la destruction de formules existantes : pour chaque formule détruite, l'entrée correspondante est libérée dans le répertoire

iii) l'édition des formules : l'utilisateur est en mesure de modifier la définition de ses formules à l'aide de l'éditeur DEDIT présent au sein de l'environnement INTERLISP-D

iv) l'inspection des formules : la définition des formules présentes dans le répertoire peut être visualisée au sein de la fenêtre interactive

Le répertoire est fort utile à l'usager, car il lui permet de conserver les formules les plus fréquemment utilisées et d'y référer par la suite à l'aide d'un simple identificateur

i) le support du gestionnaire des opérateurs

De façon similaire au point h, on retrouve un répertoire permettant de conserver des instanciations complètes des opérateurs INIT, AJOUT et ECHANGE. Ces opérateurs peuvent par la suite être introduits au sein d'une séquence d'activation particulière. Cette facilité correspond de fait à un premier pas vers la génération automatique de programmes. En effet, un programme se définit ici essentiellement comme une suite d'opérateurs correctement instanciés (figure 4.2)

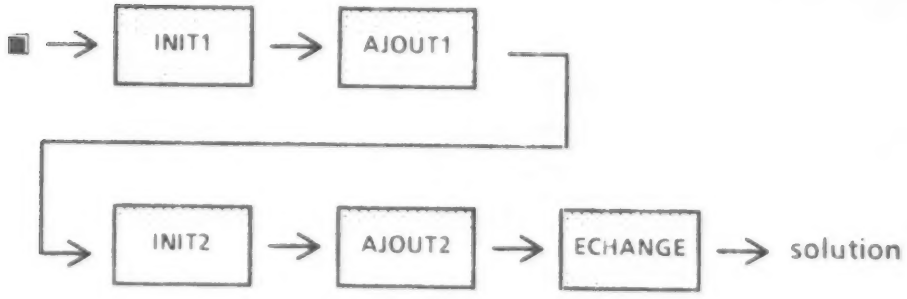


Fig. 4.2 Un exemple de 'programme'

j) le support du gestionnaire pour les critères de qualité

Dans notre système, une formule de qualité pour les routes se construit à partir d'un ensemble de critères conservés dans un répertoire. Chacun des critères est identifié de façon unique et possède les caractéristiques suivantes:

i) un critère de base (l'un des six critères mentionnés au chapitre 3)
ii) un optimum
iii) un écart maximal
iv) deux multiplicateurs d'écart
v) un ensemble de référence
vi) une courte description

L'usager se construit donc une "banque de critères" au sein de laquelle il peut puiser afin de créer les formules de qualité de son choix. Lors de la génération d'une formule, l'utilisateur doit indiquer le poids qu'il désire associer à chacun des critères retenus. Cette information est essentielle à la pondération des différents éléments présents au sein de la formule.

k) le support de formules de "défaut"

Les formules nécessaires afin de reproduire les heuristiques basées sur les méthodes d'insertion [33] sont disponibles au sein de menus spécialement conçus à cet effet. L'usager peut donc utiliser directement ces formules afin d'instancier l'heuristique générale.

l) la compilation des formules

Les formules qui servent à instancier les opérateurs de base sont compilées, ce qui permet de réduire considérablement le temps d'exécution de l'opérateur AJOUT (ce dernier s'identifiant à un mécanisme itératif au sein duquel les formules doivent constamment être réévaluées).

4.2 Quelques considérations d'ordre pratique

Avant d'aborder les autres sections, il serait bon de préciser que la notation retenue lors de la présentation des exemples sera de nature symbolique. En pratique, les formules définies par l'usager correspondent à de véritables fonctions LISP. L'utilisation du langage LISP pour la spécification des formules présente deux avantages majeurs:

i) Un usager non familier avec le langage LISP est en mesure d'utiliser une syntaxe plus "classique" par le biais de CLISP (Conversational Lisp). CLISP offre en effet certaines constructions syntaxiques similaires à celles que l'on retrouve dans les langages de programmation standards, telles une boucle for, un énoncé conditionnel if...then...else, etc.

ii) L'usager familier avec le langage LISP peut rapidement s'évader des facilités prédéfinies et construire des formules à l'aide de ses propres primitives et prédicats (cf. section 4.3). Il peut donc mettre à profit toute la puissance du langage lors de la conception des formules nécessaires à l'instanciation de l'heuristique générale.

Toutes les facilités offertes à l'utilisateur pour la définition des formules et des paramètres sont accessibles par le biais d'un menu associé à la classe RESEAU-HEURISTIQUE. Ce menu est activé suite à la sélection de l'item "ACTIVER-HEURIS" au sein du menu attaché à la fenêtre d'affichage du réseau. L'heuristique générale se trouve ainsi parfaitement intégrée à l'ensemble du système et de fait, celle-ci utilise de nombreuses facilités présentes dans l'environnement telles l'affichage des parcours, le calcul des chemins plus courts, etc.

Nous allons maintenant nous intéresser à la description des éléments prédéfinis au sein du système.

4.3 Les facilités offertes pour la définition des formules

L'utilisateur dispose d'un certain nombre de facilités lui permettant de créer les formules de son choix. Plus précisément, on retrouve :

a) les ensembles prédéfinis

E_r = ensemble des routes

E_v = ensemble des véhicules

E_{vl} = ensemble des véhicules libres (non utilisés)

E_k = ensemble des arrêts libres

E_i = ensemble des arrêts sur les routes, y inclus l'origine

E_p = ensemble des premiers arrêts visités

E_d = ensemble des derniers arrêts visités

E_e = ensemble des premiers et derniers arrêts visités ($E_p \cup E_d$)

E_s = ensemble des germes (seeds)

Il faut noter que l'interprétation de certains ensembles se trouve modifiée en cas de sélection d'une route particulière au sein de l'opérateur AJOUT . Supposant que la route r_1 est sélectionnée, nous aurons par exemple :

E_i = ensemble des arrêts sur r_1

E_s = ensemble des germes de r_1

E_p = premier arrêt visité sur r_1

E_d = dernier arrêt visité sur r_1

E_e = premier et dernier arrêts visités sur r_1

Soulignons que tous les ensembles prédéfinis sont automatiquement mis-à-jour par le système. Ainsi, un arrêt libre sera retiré de l'ensemble E_k lors de son insertion au sein d'une route.

b) les variables prédéfinies

o = l'origine

k = le dernier arrêt libre sélectionné par l'opérateur AJOUT

i = le dernier arrêt sur une route sélectionné par l'opérateur AJOUT

r = la dernière route sélectionnée par l'opérateur AJOUT

N_t = le nombre total d'arrêts

N_k = le nombre d'arrêts libres

LAMBDAcap, LAMBDAdist, LAMBDAtemps, LAMBDAfen1 et LAMBDAfen2
= les paramètres de pondération

c) les opérateurs ensemblistes

Ces opérateurs permettent d'appliquer une fonction particulière à un ensemble d'objets. En l'occurrence :

i) (**MINOBJ** <ENSEMBLE> <FONCTION>) retourne l'élément de l'ensemble pour lequel la valeur de la fonction est minimale.

ii) **(MAXOBJ** <ENSEMBLE> <FONCTION>) retourne l'élément de l'ensemble pour lequel la valeur de la fonction est maximale.

iii) **(RANDOBJ** <ENSEMBLE>) retourne un élément choisi aléatoirement au sein de l'ensemble.

iv) **(MINVAL** <ENSEMBLE> <FONCTION>) retourne la valeur minimale de la fonction dans l'ensemble.

v) **(MAXVAL** <ENSEMBLE> <FONCTION>) retourne la valeur maximale de la fonction dans l'ensemble.

vi) **(SOMME** <ENSEMBLE> <FONCTION>) retourne la somme des valeurs de la fonction dans l'ensemble.

vii) **(SOUS-ENS** <ENSEMBLE> <PREDICAT>) retourne le sous-ensemble des éléments vérifiant le prédicat.

d) Les primitives

Les concepts les plus fréquemment utilisés lors de la génération des parcours ont eté implantés dans le système sous forme de "primitives" et sont directement accessibles à l'usager. Dans ce qui suit, nous adopterons la notation suivante :

n : un noeud du réseau (arrêt ou origine)
a : un arrêt
o : l'origine
r : une route
v : un vehicule

i) les primitives d'acces

Il s'agit des primitives autorisant l'accès aux attributs des objets . Nous y retrouvons :

1) pour un noeud (arrêt ou origine)

- $ROUTE_n$: la route au sein de laquelle se trouve le noeud n
- $SUCC_n$: le successeur du noeud n si celui-ci se trouve au sein d'une route

- $PRED_n$: le prédécesseur du noeud n si celui-ci se trouve au sein d'une route

2) pour l'origine

- NV_o : le nombre de véhicules regroupés au sein de l'origine

3) pour un arrêt

- DEM_a : la demande
- $TDEM_a$: le temps alloué afin de recueillir la demande
- $TMIN_a$: la borne inférieure de la fenêtre de temps
- $TMAX_a$: la borne supérieure de la fenêtre de temps
- $TOPT_a$: le temps optimal de service
- $TARR_a$: le temps d'arrivée du véhicule à l'arrêt
- $TATT_a$: le temps d'attente avant la cueillette de la demande

4) pour un véhicule

- $CAPACITE_v$: la capacité
- $VITESSE_v$: la vitesse
- $DISTmax_v$: la distance maximale autorisée
- $TEMPSmax_v$: le temps maximal autorisé
- $ROUTE_v$: la route desservie

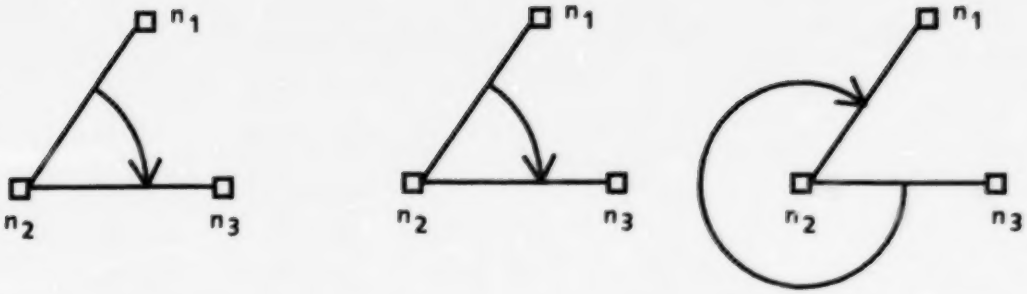
5) pour une route

- $LONGUEUR_r$: la longueur
- $COMPACITE_r$: la compacité (telle que définie en [11])
- $DEMtot_r$: la demande totale (somme des demandes aux arrêts)
- $TEMPStot_r$: le temps de parcours total
- $CARD_r$: le nombre d'arrêts au sein de la route
- $ARRETS_r$: l'ensemble des arrêts au sein de la route
- $PREMIER_r$: le premier arrêt visité
- $DERNIER_r$: le dernier arrêt visité
- $SEEDS_r$: l'ensemble des seeds ayant permis de générer la route
- $VEHICULE_r$: le véhicule desservant la route

ii) les primitives de construction des routes

Ces primitives correspondent aux concepts utilisés lors de la génération des parcours

- $ANGLE_{n_1n_2n_3}$: l'angle formé par les noeuds n_1, n_2 et n_3. Le noeud n_2 correspond ici au sommet de cet angle (voir figure 4.3a)
- $ANGLEH_{n_1n_2n_3}$: l'angle "horaire", défini dans le sens des aiguilles d'une montre (voir figure 4.3b)
- $DETOUR_{n_1n_2n_3}$: le détour qu'entraîne l'insertion du noeud n_2 entre n_1 et n_3, i.e. $DIST_{n_1n_2} + DIST_{n_2n_3} - DIST_{n_1n_3}$
- $DIST_{n_1n_2}$: la longueur du chemin plus court entre n_1 et n_2
- $DIST.EUCLID_{n_1n_2}$: la distance euclidienne entre n_1 et n_2
- $RATIO_{n_1n_2n_3}$: $(DIST_{n_1n_2} + DIST_{n_2n_3}) / DIST_{n_1n_3}$
- $RETARD_{n_1n_2n_3v}$: $DETOUR_{n_1n_2n_3} / VITESSE_v$
- $SAVING_{n_1n_2}$: $DIST_{n_1o} + DIST_{on_2} - DIST_{n_1n_2}$
- $TEMPS_{n_1n_2v}$: $DIST_{n_1n_2} / VITESSE_v$
- $NOUVE_r$: la route ou les routes créées suite à la dernière activation de l'operateur INIT



a) $ANGLE_{n_1 n_2 n_3} = ANGLE_{n_3 n_2 n_1}$

b) $ANGLEH_{n_1 n_2 n_3}$ $ANGLEH_{n_3 n_2 n_1}$

Fig. 4.3 Les notions d'angle et d'angle horaire

iii) les primitives pour le traitement des contraintes

Ces primitives sont utiles lorsque l'on désire introduire des éléments tenant compte des contraintes présentes dans un problème (ici, v = $VEHICULE_r$).

- $EXCEScap_r$: MAX (0, $DEMTOT_r$ - (LAMBDAcap * $CAPACITE_v$))
- $EXCESdist_r$: MAX (0, $LONGUEUR_r$ - (LAMBDAdist * $DISTmax_v$))
- $EXCEStemps_r$: MAX (0, $TEMPStot_r$ - (LAMBDAtemps * $TEMPSmax_v$))
- $SATURATIONcap_r$: $DEMtot_r$ / (LAMBDAcap * $CAPACITE_v$)
- $SATURATIONdist_r$: $LONGUEUR_r$ / (LAMBDAdist * $DISTmax_v$)
- $SATURATIONtemps_r$: $TEMPStot_r$ / (LAMBDAtemps * $TEMPSmax_v$)

iv) les predicats

Il s'agit de primitives retournant la valeur VRAI ou FAUX. Les prédicats s'avèrent particulièrement utiles si l'on veut identifier la nature d'un noeud ou encore déterminer l'appartenance d'un noeud à un ensemble donné

- ARRET?$_n$: le noeud n est-il un arrêt?
- DERNIER?$_a$: si l'arrêt a est inclus au sein d'une route, s'agit-il du dernier arrêt visité?
- INSERER?$_{n_1 n_2}$: est-il possible d'insérer n_2 avant ou après n_1 (dépendant de l'ordre d'insertion) et de respecter les contraintes?
- LIBRE?$_a$: l'arrêt a est-il libre?
- MEMBRE?$_{nE}$: le noeud n appartient-il à l'ensemble E?
- ORIGINE?$_n$: le noeud n correspond-t-il à l'origine?
- PREMIER?$_a$: si l'arrêt a est inclus au sein d'une route, s'agit-il du premier arrêt visité?
- VIDE?$_E$: l'ensemble E est-il vide?

e) les opérateurs mathematiques et logiques

Tous les opérateurs mathématiques et logiques usuels sont disponibles (+ , - , * , / , ET , OU , NON, etc ..). L'usager peut donc combiner à loisir les primitives presentées ci-haut.

4.4 Quelques exemples

Nous présentons ci-après quatre exemples choisis au sein de l'ensemble des algorithmes classiques qui peuvent être reproduits a l'aide de l'heuristique générale. Les operateurs INIT et AJOUT nous permettront ici de décrire chacun de ces algorithmes de construction de tournées (ou d'amas). Evidemment, il demeure possible par la suite d'améliorer la solution produite à l'aide d'une strategie d'échange appropriée. En annexe, les mêmes exemples sont repris, mais en utilisant cette fois la notation réelle en langage CLISP.

Dans tous les exemples qui suivent, nous avons :

a) une origine o

b) une flotte homogène

c) $k' \in SE_k$; $i' \in E_i$; $s' \in E_s$; $r' \in E_r$;

d) $j' = SUCC_{i'}$

Au chapitre suivant, nous verrons comment il est possible de profiter de la souplesse du système afin de résoudre un problème complexe de collecte du courrier présentant des contraintes multiples.

Exemple 1. "Farthest insertion"

INIT

N ← 1;

SE_k ← E_k;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← **(MAXOBJ SE_k $DIST_{ok'}$)**;

(i.e $CMO_{k'}$ = $DIST_{ok'}$ et on sélectionne l'arrêt k' possédant le CMO maximal)

v ← **(RANDOBJ E_{vl})** ; (i.e sélection aléatoire du véhicule)

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N-1;

AJOUT

SE_k ← E_k

Tant que SE_k ≠ Ø faire

r_1 ← **NIL**;

k ← **(MAXOBJ SE_k (MINVAL E_i $DIST_{i'k'}$))**;

(i.e $CMO_{k'}$ = (MINVAL E_i $DIST_{i'k'}$) et on sélectionne l'arrêt k' possédant le CMO maximal)

r_2 ← **NIL**;

i ← **(MINOBJ E_i $DETOUR_{i'k_j'}$)**,

(i.e $CMO_{i'}$ = $DETOUR_{i'k_j'}$ et on sélectionne l'arrêt i' possédant le CMO minimal)

si **NIL** verifie alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k **après** i

sinon

traiter k selon **mode libre**,

retirer k de SE_k;

Contraintes : aucune

Exemple 2. "Nearest neighbor"

INIT

N ← 1;
SE_k ← E_k;
Tant que N > 0 et qu'il reste des véhicules disponibles faire
    GERMES ← (MINOBJ SE_k $DIST_{ok'}$);
    v ← (RANDOBJ E_{vl});
    créer une route où les GERMES sont desservis par le véhicule v;
    SE_k ← SE_k - GERMES;
    N ← N - 1;

AJOUT

SE_k ← E_k;
Tant que SE_k ≠ Ø faire
    r_1 ← $NOUV_{E_r}$;
    k ← (MINOBJ SE_k $DIST_{DERNIER_{r_1}k'}$);
    r_2 ← NIL;
    i ← $DERNIER_{r_1}$;
    si NIL vérifié alors sortir de la boucle et terminer;
    si l'insertion de k respecte les contraintes alors
        insérer k après i
    sinon
        traiter k selon mode libre;
    retirer k de SE_k;

Contraintes : aucune

Exemple 3. Algorithme de balayage

On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'il n'y ait plus d'arrêts libres

INIT

N ← **1**;

SE_k ← E_k;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← (si **VIDE?**$_{E_r}$ alors
(MAXOBJ SE_k **DIST**$_{ok'}$)
sinon
(MINOBJ SE_k **(MINVAL** E_s **ANGLEH**$_{s'ok'}$)))

v ← **(RANDOBJ** E_{vl}**)**;

créer une route où les GERMES sont desservis par le véhicule v,

SE_k ← SE_k - GERMES;

N ← N-1;

AJOUT

SE_k ← E_k;

Tant que SE_k ≠ ∅ faire

r_1 ← **NOUV**$_{E_r}$;

k ← **(MINOBJ** SE_k **ANGLEH**$_{DERNIER_{r_1}ok'}$ **)**;

r_2 ← **NIL**;

i ← **DERNIER**$_{r_1}$;

si **NIL** vérifié alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k **après** i

sinon

traiter k selon **mode interruption**;

retirer k de SE_k;

Contraintes : LAMBDAcap = **1.0**

Exemple 4. Algorithme de Ferland et Rousseau

Phase 1. On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'on ait ⌈demande totale / capacité⌉ routes.

INIT

$N \leftarrow 1$;
$SE_k \leftarrow E_k$;
Tant que N > 0 et qu'il reste des véhicules disponibles faire
    GERMES $\leftarrow$ **(MAXOBJ** SE_k $DIST_{k'o}$**)**;
    v $\leftarrow$ **(RANDOBJ** E_{vl}**)**;
    créer une route où les GERMES sont desservis par le véhicule v;
    $SE_k \leftarrow SE_k$ - GERMES;
    $N \leftarrow N-1$;

AJOUT

$SE_k \leftarrow E_k$;
Tant que $SE_k \neq \emptyset$ faire
    $r_1 \leftarrow$ **NOUV**$_{E_r}$;
    k $\leftarrow$ **(MINOBJ** SE_k (formule de Ferland & Rousseau)
        (MINVAL E_s
            (si $DIST_{s'k'} \leq DIST_{s'o}$ **alors**
                $DIST_{s'k'}^2 * (1 + LAMBDA *$
                $(1 - \frac{DIST_{s'k'}^2 - DIST_{k'o}^2 + DIST_{s'o}^2}{2 * DIST_{s'k'} * DIST_{s'o}}))$
            sinon $DIST_{s'k'}^2 * (1 + LAMBDA * (DIST_{k'o} / DIST_{s'k'})))))$;
    $r_2 \leftarrow$ **NIL**;
    i $\leftarrow$ **(RANDOBJ** E_i**)**;
    si **NIL** vérifié alors sortir de la boucle et terminer;
    si l'insertion de k respecte les contraintes alors
        insérer k **après** i
    sinon
        traiter k selon **mode interruption**;
    retirer k de SE_k;

Contraintes : LAMBDAcap = 1.0

Exemple 4 Algorithme de Ferland et Rousseau

Phase 2. On active l'opérateur AJOUT de façon à introduire les arrêts qui demeurent libres suite à la phase 1

AJOUT

$SE_k \leftarrow E_k$;
Tant que $SE_k \neq \emptyset$ faire

$r_1 \leftarrow$ **(MINOBJ** E_r $DEMtot_{r'}$**)**,

$k \leftarrow$ **(MINOBJ** SE_k
(MINVAL E_i
(si $DIST_{i'k'} \leq DIST_{i'o}$ **alors**

$$DIST_{i'k'}^2 * (1 + LAMBDA * \frac{(1 - DIST_{i'k'}^2 - DIST_{k'o}^2 + DIST_{i'o}^2)}{2 * DIST_{i'k'} * DIST_{i'o}})$$

sinon $DIST_{i'k'}^2 * (1 + LAMBDA * (DIST_{k'o} / DIST_{i'k'}))))$;

$r_2 \leftarrow$ **NIL**,

$i \leftarrow$ **(RANDOBJ** E_i**)**;

si **NIL** vérifie alors sortir de la boucle et terminer,

si l'insertion de k respecte les contraintes alors

insérer k **après** i

sinon

traiter k selon **mode libre**,

retirer k de SE_k;

Contraintes : aucune

Chapitre 5. Une application concrète de l'heuristique générale

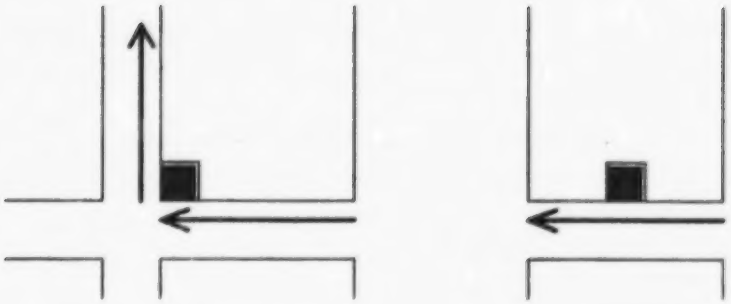
Afin d'illustrer l'intérêt de l'heuristique générale pour la génération de tournées, nous nous sommes attaqués à un problème réel de collecte du courrier par des véhicules postaux. Cet exemple démontrera l'utilité de notre système lors du développement de nouvelles approches algorithmiques face à un problème pour lequel les stratégies de résolution ne sont pas encore bien définies.

5.1 Présentation du problème

Les données du problème proviennent de Postes Canada, un organisme gouvernemental qui a fait appel au Centre de Recherche sur les Transports afin de développer un algorithme susceptible de produire des solutions supérieures à celles des approches traditionnelles non informatisées. Ces efforts de recherche ont mené à la création d'un logiciel connu sous le nom de POST9. La disponibilité de cet algorithme spécialisé nous fournit maintenant une base de comparaison afin d'évaluer la performance de l'heuristique générale.

Le réseau desservi correspond à la zone Bourassa, située au nord de l'île de Montréal, au sein de laquelle nous avons localisé 37 succursales postales (voir la figure 5.2, à la fin du chapitre). L'ensemble des véhicules, partant de l'origine, doit ici desservir la totalité des succursales ou arrêts avant de revenir au point de départ pour y déposer le chargement.

Les contraintes présentes au sein de ce problème sont multiples. On retrouve tout d'abord des contraintes touchant à la structure même du réseau sous-jacent, telles les sens uniques et les interdictions de virage. Ensuite, la succursale desservie se doit d'être située à la droite du véhicule, ce qui introduit des considérations d'accès sélectif. Une succursale ne pourra ainsi être atteinte qu'en empruntant un ou deux arcs prédéterminés (figure 5.1). Enfin, on associe à chacune des succursales postales une fenêtre de temps et à chacun des véhicules un temps maximal de parcours autorisé.



(a) deux arcs (coin) (b) un seul arc

Fig. 5.1 Accès sélectif à un arrêt

Pour le type de problème considéré, la borne supérieure de chacune des fenêtres de temps constitue une contrainte inélastique, car un véhicule ne peut se permettre d'arriver après la fermeture d'une succursale. Par contre, le non respect d'une borne inférieure est toléré et n'entraîne qu'un temps d'attente supplémentaire jusqu'à ce que le courrier soit disponible. Notons enfin que le nombre total de véhicules à utiliser n'est pas fixé a priori et qu'il n'y a pas de contraintes de capacité ou de distance maximale autorisée.

Lors de la résolution, la vitesse moyenne des véhicules a été fixée à 35 kilomètres / heure avec un temps maximal de parcours de 120 minutes.

5.2 Un algorithme de résolution spécialisé (POST9)

POST9 est un algorithme spécialisé pour les problèmes de collecte du courrier qui permet d'obtenir des solutions supérieures à celles produites manuellement par des humains. Cet algorithme se divise en deux phases principales: d'abord une phase de génération des amas selon la technique de Ferland & Rousseau [11], puis une phase d'échange introduisant une forte pénalité en cas de non respect des fenêtres de temps. Au cours de la phase de génération des amas, seul le concept de localité est utilisé et il n'est aucunement tenu compte des contraintes horaires. C'est essentiellement le processus d'échange qui s'efforce de générer une solution réalisable de longueur minimale. Divers ajustements sont possibles, tels l'ajout de nouvelles routes, si la phase d'échange ne permet pas de produire une telle solution réalisable.

A la fin du processus de résolution, une transformation triviale est appliquée à la solution obtenue de façon à minimiser les temps d'attente. On obtient ce résultat en identifiant,

pour chaque véhicule, le départ le plus tardif assurant le respect de la borne supérieure pour chacune des fenêtres de temps le long du parcours.

La solution minimisant la distance totale parcourue par les véhicules est présentée à la figure 5.3. Pour chacune des routes, on retrouve la distance, le temps de parcours et les numéros des arrêts visités. De plus, on présente pour chaque arrêt le temps d'arrivée et de départ du véhicule ainsi que la fenêtre de temps. Les unités de temps s'expriment en minutes et la fenêtre est représentée sous la forme (borne inférieure, borne supérieure). Pour la solution POST9, la distance totale est de 46.80 kms et le temps de parcours est de 261 minutes, incluant 32 minutes de temps d'attente. Ce résultat nous servira maintenant de base de comparaison afin d'évaluer les bénéfices liés à l'application de l'heuristique générale.

5.3 L'heuristique générale

Nous avons cherché ici à profiter de la souplesse offerte par notre système afin de générer diverses alternatives de résolution. Au cours de cet exercice, nous nous sommes efforcés d'éviter les approches trop sophistiquées afin d'identifier clairement certains principes de base menant à des résultats intéressants. Nous présentons ci-après trois approches différentes qui ont toutes produit une solution supérieure à celle de POST9.

5.3.1 Approche séquentielle 1 (SEQ1)

Cette première approche, où les routes sont générées de façon séquentielle, s'efforce de respecter à tout moment les contraintes horaires. Une solution réalisable est donc disponible dès avant l'application des procédures d'échange.

Le principe à la base de la stratégie de résolution est relativement simple: au cours de la phase de génération des amas (technique de Ferland et Rousseau), chacun des nouveaux arrêts est inséré selon le principe du "plus petit détour". Cette insertion se limite aux endroits assurant le respect de la borne supérieure de la fenêtre de temps à chacun des arrêts et le respect du temps maximal autorisé pour le véhicule. S'il n'existe pas d'endroit réalisable, le nouvel arrêt demeure libre et sera pris en charge par un autre véhicule.

Une telle approche présente deux avantages majeurs. Tout d'abord, puisque l'on considère explicitement les contraintes horaires lors de la phase de génération des amas, une solution réalisable est disponible en tout temps. De plus, comme le processus d'insertion obéit au principe du "plus petit détour", les routes générées sont de longueur raisonnable, ce qui limite le risque d'aboutir dans des culs-de-sac inintéressants lors de l'application des procédures d'échange.

La meilleure solution a été obtenue en n'autorisant aucun véhicule à desservir plus du tiers des succursales. La solution obtenue à ce niveau est présentée à la figure 5.4b. Par la suite, deux stratégies d'échange distinctes ont été appliquées. La première se contente d'appliquer un 2-opt individuel à chacune des routes avant de compléter avec un 2-opt généralisé à l'ensemble des routes. La seconde technique introduit des échanges de type 3-opt dans la séquence d'activation, soit: un 2-opt individuel, un 3-opt individuel, un 2-opt général et enfin un 3-opt individuel. Dans chacun des cas, le critère de qualité retenu correspond à la distance totale parcourue par les véhicules.

Les solutions obtenues sont présentées aux figures 5.4c et 5.4d. Evidemment, la seconde stratégie introduisant des échanges de type 3-opt permet d'obtenir un résultat supérieur. Dans les deux cas toutefois, on note une amélioration substantielle par rapport à la solution POST9: la première solution assure une économie de 5.5 kms (amélioration de l'ordre de 11%) alors que la seconde assure une économie de 8 4 kms (amélioration de l'ordre de 17%). Dans ce dernier cas, on retranche également 8 minutes au niveau des temps d'attente, mais il s'agit d'un bénéfice non anticipé puiqu'en aucune façon il n'a été tenu compte de ce facteur au cours du processus de résolution. On doit toutefois observer que le seul fait de réduire les distances et les temps de déplacement accorde une plus grande liberté à la procédure de minimisation des temps d'attente. Celle-ci dispose alors d'écarts plus considérables entre le temps d'arrivée du véhicule et la borne supérieure de la fenêtre de temps à chacun des arrêts, ce qui permet de retarder encore davantage le moment du départ.

L'approche SEQ1 que nous venons de décrire est celle qui a généré les résultats les plus intéressants. L'instanciation exacte des opérateurs INIT et AJOUT ayant permis d'obtenir de telles solutions est présentée à la figure 5.4a.

5.3.2 Approche séquentielle 2 (SEQ2)

Nous avons voulu ici nous rapprocher davantage de l'approche POST9 en ignorant tout d'abord les contraintes horaires. La seule modification par rapport à POST9 consiste à ordonnancer les arrêts au sein des amas d'une façon particulière: les arrêts sont ici disposés en ordre croissant selon la borne supérieure de leur fenêtre de temps. L'utilisation d'un tel principe facilite par la suite la génération d'une solution réalisable, les arrêts à visiter le plus tôt étant déjà situés au début des parcours.

Une seule application d'un 2-opt individuel sur chacune des routes a d'ailleurs permis de produire une telle solution réalisable. Le critère de qualité reposait ici sur le temps d'arrivée du véhicule à chacun des arrêts. L'intervalle de variation pour l'optimum ayant été fixé à

[0 , borne supérieure de la fenêtre de temps], toute arrivée excédant la borne supérieure se trouvait à introduire une pénalité d'ordre quadratique (cf. chapitre 3).

Par la suite, la séquence d'échanges incluant les 3-opt a été activée en retenant pour critère de qualité la distance totale parcourue par les véhicules. Le résultat obtenu à l'aide de cette approche est présenté à la figure 5.5b. Cette solution représente une économie de 3 kms par rapport à la solution POST9, soit une réduction de l'ordre de 6%.

5.3.3 Approche parallèle (PAR1)

Nous avons profité de la souplesse offerte par l'heuristique générale afin de développer une stratégie basée sur la construction parallèle de routes selon un principe de localité temporelle. Pour ce faire, les arrêts ont été divisés en trois sous-ensembles ou catégories distinctes:

a) catégorie 1 : borne supérieure de la fenêtre de temps ≤ 60

b) catégorie 2 : borne supérieure de la fenêtre de temps ≤ 90

c) catégorie 3 : borne supérieure de la fenêtre de temps ≤ 120

Chaque catégorie a été traitée à tour de rôle à l'aide d'une stratégie d'insertion fondée sur des principes identiques à ceux de SEQ2, soit: recours au "plus petit détour" dans un contexte de respect des contraintes horaires. Après l'application des procédures d'échange, nous avons obtenu le résultat présenté à la figure 5.6b. Il s'agit ici d'une économie de 4.3 kms par rapport à la solution POST9, soit une réduction de l'ordre de 8%.

5.4 Conclusion

La qualité de la solution obtenue à l'aide de l'approche SEQ1 met en évidence un défaut majeur de l'algorithme POST9: au cours de la phase de génération des amas, aucune considération n'est accordée aux contraintes horaires. Ceci conduit à la génération de parcours non réalisables et le travail de la procédure d'échange consiste davantage à s'orienter vers une solution réalisable qu'à produire une solution de bonne qualité.

A la lumière de cette expérience, il apparaît qu'une légère modification à l'algorithme POST9 afin d'assurer le respect des contraintes horaires lors de la phase de génération des amas, devrait permettre d'améliorer grandement la qualité des solutions obtenues. On voit donc, grâce à cet exemple, comment l'heuristique générale peut être utilisée afin de raffiner et améliorer des algorithmes existants.

5.5 Observations complémentaires

Comme on peut le constater, le recours à l'heuristique générale face à un problème particulier autorise le développement d'approches de résolution menant à des résultats supérieurs à ceux d'un algorithme spécialisé. La flexibilité offerte par cette heuristique constitue ici un avantage majeur, puisqu'elle permet d'aborder un problème selon des perspectives diverses afin d'identifier les avenues de résolution les plus prometteuses. Il est également facile d'ajouter de nouveaux éléments au sein de la séquence d'activation des opérateurs de base et d'en observer les effets. C'est ce que nous avons fait en particulier lorsqu'il s'est agi d'introduire les échanges de type 3-opt. Il a suffi pour cela de créer deux nouvelles instanciations de l'opérateur ECHANGE et de les insérer à l'endroit approprié dans la séquence d'activation.

Sur le plan strictement pratique, le problème majeur de l'implantation actuelle provient de la vitesse d'exécution qui laisse à désirer dans le cas de l'opérateur d'échange. Ainsi, pour le problème que nous venons d'étudier, la procédure 2-opt appliquée à l'ensemble des routes (38 noeuds) requiert environ 10 minutes en temps d'exécution. Ce problème provient de trois sources distinctes:

i) La machine Lisp a été conçue spécialement pour la manipulation de symboles et se révèle moins efficace lors de calculs numériques.

ii) Le souci de généralité nous empêche de réaliser une implantation véritablement performante. En effet, six critères distincts sont susceptibles d'être introduits au sein de la formule de qualité lors de l'application de la procédure d'échange. Ceci prévient le développement d'une implantation fondée uniquement sur la minimisation des distances. Il nous faut donc modifier les routes pour chacun des échanges considérés, de façon à évaluer correctement la qualité de la nouvelle solution. Un tel processus est évidemment beaucoup plus lent que la simple évaluation de la différence entre la longueur des arcs ajoutés et retranchés.

iii) Le programme ALTO étant de dimension considérable (environ 12,000 lignes de code), nous n'avons pas cherché à améliorer la performance au détriment de la qualité de la programmation. Une fois la phase de développement completée, il pourrait toutefois s'avérer profitable de consacrer certains efforts à la mise au point d'une version plus performante. Il reste neanmoins que l'objectif visé n'était pas de produire un système operationnel capable de répondre à des applications en temps réel, mais bien de fournir un outil favorisant le développement d'algorithmes de résolution.

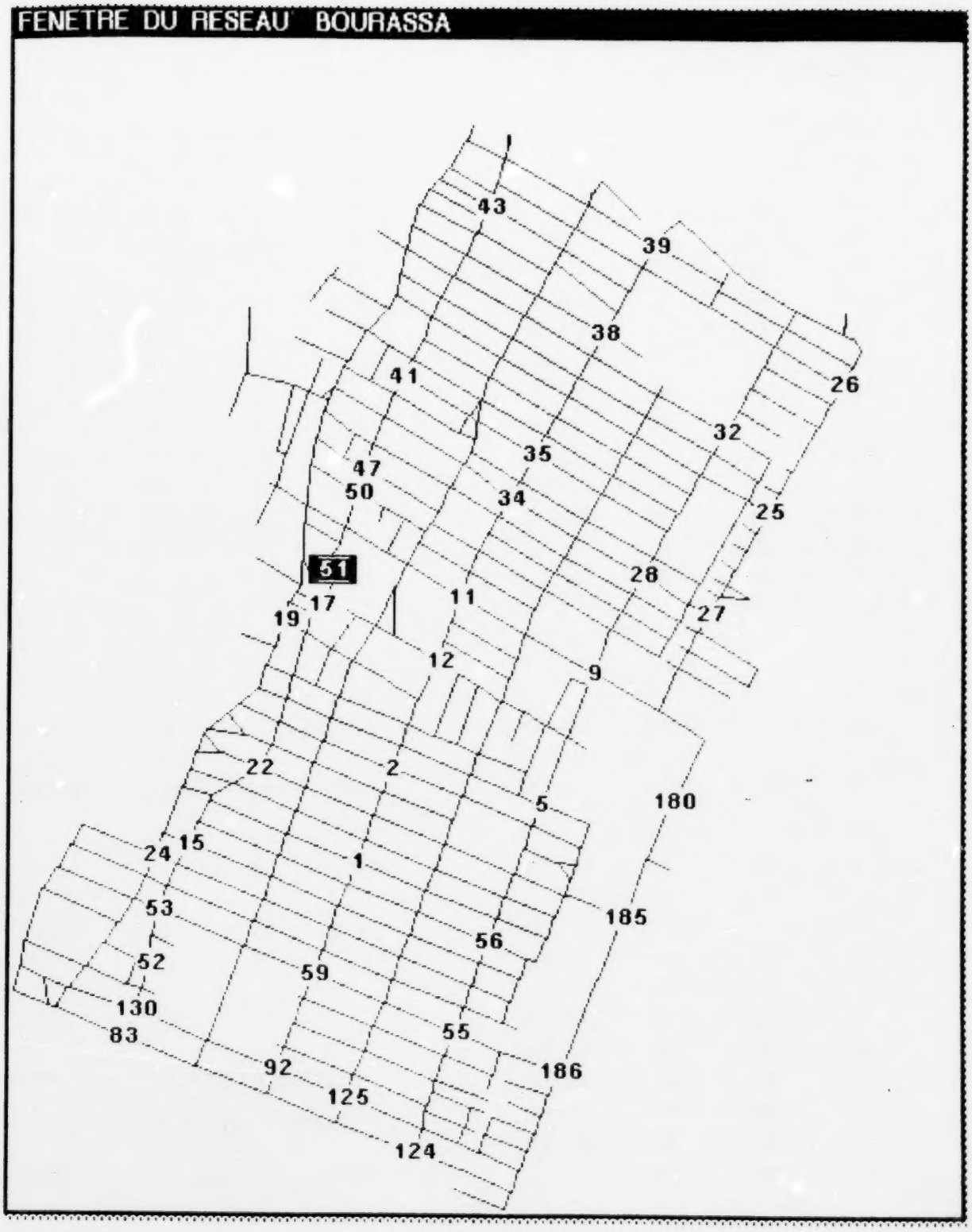


Fig. 5.2 La zone Bourassa

SOLUTION POST9

ROUTE 1 17.32 kms 78 min

ARRET	ARRIVEE	DEPART	FENETRE
51		26	
53	32	33	(5 , 55)
59	38	39	(10 , 50)
56	43	45	(35 , 55)
125	50	51	(50 , 80)
124	55	56	(15 , 65)
92	60	61	(60 , 90)
83	65	66	(60 , 85)
52	69	70	(30 , 120)
15	73	74	(60 , 90)
1	78	80	(60 , 105)
55	84	85	(60 , 120)
186	90	91	(60 , 90)
130	97	98	(60 , 120)
51	104		

ROUTE 2 14.51 kms 69 min

ARRET	ARRIVEE	DEPART	FENETRE
51		44	
180	50	51	(30 , 60)
25	56	58	(0 , 105)
28	62	64	(45 , 70)
34	67	68	(60 , 80)
35	72	73	(60 , 80)
39	79	80	(60 , 88)
26	85	86	(60 , 120)
32	90	91	(60 , 120)
9	95	96	(65 , 95)
185	101	102	(60 , 105)
2	107	108	(90 , 120)
51	113		

ROUTE 3 11.18 kms 61 min

ARRET	ARRIVEE	DEPART	FENETRE
51		37	
11	40	41	(0 , 90)
12	45	47	(15 , 45)
19	51	52	(50 , 70)
47	56	61	(60 , 85)
41	65	66	(30 , 90)
43	70	72	(60 , 70)
50	78	79	(65 , 90)
22	83	84	(60 , 120)
24	88	89	(70 , 90)
17	93	94	(45 , 95)
51	98		

ROUTE 4 6.79 kms 53 min

ARRET	ARRIVEE	DEPART	FENETRE
51		24	
38	30	31	(0 , 30)
27	37	61	(60 , 90)
5	65	71	(70 , 90)
51	77		

DISTANCE TOTALE : 49.80 kms

TEMPS TOTAL : 261 min.*

* incluant 32 min. d'attente

Fig. 5.3 Algorithme POST9

SEQ1 On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'il n'y ait plus d'arrêts libres.

INIT

$N \leftarrow 1$;
$SE_k \leftarrow E_k$;
Tant que N > 0 et qu'il reste des véhicules disponibles faire
    GERMES $\leftarrow$ **(MAXOBJ** SE_k $DIST_{k'o}$**)**;
    v $\leftarrow$ **(RANDOBJ** E_{vl}**)**;
    créer une route où les GERMES sont desservis par le véhicule v;
    $SE_k \leftarrow SE_k$ - GERMES;
    $N \leftarrow N-1$;

AJOUT

$SE_k \leftarrow E_k$;
Tant que $SE_k \neq \emptyset$ faire
    $r_1 \leftarrow$ **NOUV**E_r;
    k $\leftarrow$ **(MINOBJ** SE_k (formule de Ferland & Rousseau avec LAMBDA = 1.0)
        (MINVAL E_s
            (si $DIST_{s'k'} \leq DIST_{s'o}$ **alors**
                $DIST_{s'k'}^2 \cdot (1 + LAMBDA \cdot (1 - \frac{DIST_{s'k'}^2 - DIST_{k'o}^2 + DIST_{s'o}^2}{2 \cdot DIST_{s'k'} \cdot DIST_{s'o}}))$
            sinon $DIST_{s'k'}^2 \cdot (1 + LAMBDA \cdot (DIST_{k'o} / DIST_{s'k'}))$))));
    $r_2 \leftarrow$ **NIL**;
    i $\leftarrow$ **(MINOBJ (SOUS-ENS** E_i **INSERER?**$_{i'k}$**)** $DETOUR_{i'kj'}$**)**;
    si $CARD_{r_1} \geq N_t / 3$ vérifié alors sortir de la boucle et terminer;
    si l'insertion de k respecte les contraintes alors
        inserer k **après** i
    sinon
        traiter k selon **mode libre**;
    retirer k de SE_k;

Contraintes : LAMBDAtemps = LAMBDAfen2 = **1.0**

Fig. 5.4 (a) Algorithme SEQ1

SOLUTION SEQ1 (avant échange)

ROUTE 1 21.96 kms 90 min

ARRET	ARRIVEE	DEPART	FENETRE
51		21	
12	25	27	(15 , 45)
180	33	34	(30 , 60)
25	39	41	(0 , 105)
28	45	47	(45 , 70)
19	53	54	(50 , 70)
11	58	59	(0 , 90)
34	64	65	(60 , 80)
5	70	71	(70 , 90)
186	76	77	(60 , 90)
24	84	85	(70 , 90)
22	90	91	(60 , 120)
17	95	96	(45 , 95)
185	103	104	(60 , 105)
51	111		

ROUTE 2 16.08 kms 83 min

ARRET	ARRIVEE	DEPART	FENETRE
51		31	
53	37	38	(5 , 55)
56	43	45	(35 , 55)
59	50	51	(10 , 50)
92	56	61	(60 , 90)
124	65	66	(15 , 65)
125	70	71	(50 , 80)
55	74	75	(60 , 120)
83	81	82	(60 , 85)
52	85	86	(30 , 120)
15	89	90	(60 , 90)
130	94	95	(60 , 120)
1	101	103	(60 , 105)
2	108	109	(90 , 120)
51	114		

ROUTE 3 10.89 kms 72 min

ARRET	ARRIVEE	DEPART	FENETRE
51		24	
38	30	31	(0 , 30)
32	37	61	(60 , 120)
26	65	66	(60 , 120)
27	70	71	(60 , 90)
9	75	76	(65 , 95)
35	80	81	(60 , 80)
47	85	86	(60 , 85)
41	90	91	(30 , 90)
51	96		

ROUTE 4 5.10 kms 24 min

ARRET	ARRIVEE	DEPART	FENETRE
51		65	
43	70	72	(60 , 70)
39	77	78	(60 , 88)
50	84	85	(65 , 90)
51	89		

DISTANCE TOTALE : 54.03 kms

TEMPS TOTAL : 269 min.*

* incluant 27 min. d'attente

Fig. 5.4 (b) Algorithme SEQ1

SOLUTION SEQ1 (avec 2-opt)

ROUTE 1 14.49 kms 65 min

ARRET	ARRIVEE	DEPART	FENETRE
51		41	
180	47	48	(30 , 60)
25	53	55	(0 , 105)
34	60	61	(60 , 80)
50	65	66	(65 , 90)
43	70	72	(60 , 70)
39	77	78	(60 , 88)
32	83	84	(60 , 120)
9	88	89	(65 , 95)
185	94	95	(60 , 105)
2	100	101	(90 , 120)
51	106		

ROUTE 2 12.03 kms 68 min

ARRET	ARRIVEE	DEPART	FENETRE
51		31	
53	37	38	(5 , 55)
56	43	45	(35 , 55)
59	50	51	(10 , 50)
92	56	61	(60 , 90)
124	65	66	(15 , 65)
125	70	71	(50 , 80)
55	74	75	(60 , 120)
83	81	82	(60 , 85)
52	85	86	(30 , 120)
15	89	90	(60 , 90)
17	94	95	(45 , 95)
51	99		

ROUTE 3 7.39 kms 70 min

ARRET	ARRIVEE	DEPART	FENETRE
51		19	
12	23	25	(15 , 45)
38	30	31	(0 , 30)
26	36	61	(60 , 120)
27	65	66	(60 , 90)
28	69	71	(45 , 70)
35	75	76	(60 , 80)
41	79	80	(30 , 90)
47	84	85	(60 , 85)
51	89		

ROUTE 4 10.41 kms 50 min

ARRET	ARRIVEE	DEPART	FENETRE
51		57	
19	61	62	(50 , 70)
11	66	67	(0 , 90)
5	71	72	(70 , 90)
186	77	78	(60 , 90)
1	84	86	(60 , 105)
24	90	91	(70 , 90)
130	95	96	(60 , 120)
22	101	102	(60 , 120)
51	107		

DISTANCE TOTALE : 44.32 kms

TEMPS TOTAL : 253 min.*

* incluant 28 min. d'attente

Fig. 5.4 (c) Algorithme SEQ1

SOLUTION SEQ1 (avec 2-opt et 3-opt)

ROUTE 1 12.87 kms 64 min

ARRET	ARRIVEE	DEPART	FENETRE
51		50	
11	53	54	(0 , 90)
180	59	60	(30 , 60)
185	63	64	(60 , 105)
186	67	68	(60 , 90)
25	76	78	(0 , 105)
32	81	82	(60 , 120)
9	86	87	(65 , 95)
5	90	91	(70 , 90)
1	96	98	(60 , 105)
2	103	104	(90 , 120)
22	108	109	(60 , 120)
51	114		

ROUTE 2 13.37 kms 72 min

ARRET	ARRIVEE	DEPART	FENETRE
51		28	
53	34	35	(5 , 55)
56	40	42	(35 , 55)
59	47	48	(10 , 50)
124	52	53	(15 , 65)
125	57	58	(50 , 80)
55	61	62	(60 , 120)
92	67	68	(60 , 90)
83	72	73	(60 , 85)
130	77	78	(60 , 120)
52	82	83	(30 , 120)
15	86	87	(60 , 90)
24	90	91	(70 , 90)
17	95	96	(45 , 95)
51	100		

ROUTE 3 8.72 kms 75 min

ARRET	ARRIVEE	DEPART	FENETRE
51		19	
12	23	25	(15 , 45)
38	30	31	(0 , 30)
26	36	61	(60 , 120)
27	65	66	(60 , 90)
28	69	71	(45 , 70)
34	74	75	(60 , 80)
35	79	80	(60 , 80)
47	84	85	(60 , 85)
50	89	90	(65 , 90)
51	94		

ROUTE 4 6.42 kms 29 min

ARRET	ARRIVEE	DEPART	FENETRE
51		59	
19	63	64	(50 , 70)
43	70	72	(60 , 70)
39	77	78	(60 , 88)
41	82	83	(30 , 90)
51	88		

DISTANCE TOTALE : 41.38 kms

TEMPS TOTAL : 240 min.*

* incluant 24 min. d'attente

Fig. 5.4 (d) Algorithme SEQ1

SEQ2. On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'il n'y ait plus d'arrêts libres

INIT

$N \leftarrow 1$;
$SE_k \leftarrow E_k$;
Tant que $N > 0$ et qu'il reste des véhicules disponibles faire
    GERMES $\leftarrow$ **(MAXOBJ** SE_k $DIST_{k'o}$**)**;
    v $\leftarrow$ **(RANDOBJ** E_{vl}**)**;
    créer une route où les GERMES sont desservis par le véhicule v;
    $SE_k \leftarrow SE_k$ - GERMES;
    $N \leftarrow N-1$;

AJOUT

$SE_k \leftarrow E_k$;
Tant que $SE_k \neq \emptyset$ faire
    $r_1 \leftarrow$ **NOUV**E_r;
    k $\leftarrow$ **(MINOBJ** SE_k (formule de Ferland & Rousseau avec LAMBDA = 1.0)
        (MINVAL E_s
            (si $DIST_{s'k'} \leq DIST_{s'o}$ **alors**
                $DIST_{s'k'}^2 \cdot (1 + LAMBDA \cdot (1 - \frac{DIST_{s'k'}^2 - DIST_{k'o}^2 + DIST_{s'o}^2}{2 \cdot DIST_{s'k'} \cdot DIST_{s'o}}))$
            sinon $DIST_{s'k'}^2 \cdot (1 + LAMBDA \cdot (DIST_{k'o} / DIST_{s'k'}))$**)))**;
    $r_2 \leftarrow$ **NIL**;
    i $\leftarrow$ **(MAXOBJ (SOUS-ENS** E_i $TMAX_{i'} \leq TMAX_k$**)** $TMAX_{i'}$**)**; $TMAX_o = 0$
    si $CARD_{r_1} \geq N_t / 4$ vérifié alors sortir de la boucle et terminer;
    si l'insertion de k respecte les contraintes alors
        insérer k **après** i
    sinon
        traiter k selon **mode libre**;
    retirer k de SE_k;

Contraintes : aucune

Fig. 5.5 (a) Algorithme SEQ2

SOLUTION SEQ2 (avec 2-opt et 3-opt)

ROUTE 1 15.03 kms 94 min

ARRET	ARRIVEE	DEPART	FENETRE
51		17	
25	24	26	(0 , 105)
38	30	31	(0 , 30)
26	36	61	(60 , 120)
27	65	66	(60 , 90)
28	69	71	(45 , 70)
35	75	76	(60 , 80)
41	79	80	(30 , 90)
32	86	87	(60 , 120)
9	91	92	(65 , 95)
185	97	98	(60 , 105)
1	103	105	(60 , 105)
51	111		

ROUTE 2 12.53 kms 69 min

ARRET	ARRIVEE	DEPART	FENETRE
51		28	
11	31	32	(0 , 90)
180	37	38	(30 , 60)
56	43	45	(35 , 55)
59	50	51	(10 , 50)
124	55	56	(15 , 65)
125	60	61	(50 , 80)
55	64	65	(60 , 120)
92	70	71	(60 , 90)
83	75	76	(60 , 85)
52	79	80	(30 , 120)
15	83	84	(60 , 90)
24	87	88	(70 , 90)
17	92	93	(45 , 95)
51	97		

ROUTE 3 11.41 kms 49 min

ARRET	ARRIVEE	DEPART	FENETRE
51		49	
53	55	56	(5 , 55)
130	59	61	(60 , 120)
19	67	68	(50 , 70)
22	72	73	(60 , 120)
186	79	80	(60 , 90)
5	87	88	(70 , 90)
2	92	93	(90 , 120)
51	98		

ROUTE 4 7.84 kms 48 min

ARRET	ARRIVEE	DEPART	FENETRE
51		41	
12	45	47	(15, 45)
34	52	61	(60 , 80)
43	66	68	(60 , 70)
39	73	74	(60 , 88)
50	80	81	(65 , 90)
47	84	85	(60 , 85)
51	89		

DISTANCE TOTALE : 46.81 kms

TEMPS TOTAL : 260 min.*

* incluant 33 min. d'attente

Fig. 5.5 (b) Algorithme SEQ2

PAR1. On active l'opérateur INIT, puis l'opérateur AJOUT avec respectivement LAMBDA = 60, 90 et 120.

INIT

N ← **4**;

SE_k ← **E_k**;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← **(MINOBJ SE_k $TMAX_{k'}$)**;

v ← **(RANDOBJ E_{vl})**;

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N-1;

AJOUT

SE_k ← **(SOUS-ENS E_k $TMAX_{k'}$ ≤ LAMBDA)**

Tant que SE_k ≠ ∅ faire

r_1 ← **NIL**;

k ← **(MINOBJ SE_k $TMAX_{k'}$)**;

r_2 ← **NIL**;

i ← **(MINOBJ (SOUS-ENS E_i INSERER?$_{i'k}$) DETOUR$_{i'kj'}$)**;

si **NIL** vérifié alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k après i

sinon

traiter k selon **mode libre**;

retirer k de SE_k;

Contraintes : LAMBDAtemps = LAMBDAfen2 = 1.0

Fig. 5.6 (a) Algorithme PAR1

SOLUTION PAR1 (avec 2-opt et 3-opt)

ROUTE 1 16.54 kms 96 min

ARRET	ARRIVEE	DEPART	FENETRE
51		17	
25	24	26	(0 , 105)
38	30	31	(0 , 30)
41	36	37	(30 , 90)
43	41	62	(60 , 70)
39	67	68	(60 , 88)
26	73	74	(60 , 120)
32	78	79	(60 , 120)
27	84	85	(60 , 90)
5	89	90	(70 , 90)
9	95	96	(65 , 95)
185	101	102	(60 , 105)
2	107	108	(90 , 120)
51	113		

ROUTE 2 14.83 kms 70 min

ARRET	ARRIVEE	DEPART	FENETRE
51		24	
11	27	28	(0 , 90)
180	33	34	(30 , 60)
59	40	41	(10 , 50)
124	45	46	(15 , 65)
125	50	51	(50 , 80)
56	55	57	(35 , 55)
186	61	62	(60 , 90)
28	69	71	(45 , 70)
34	74	75	(60 , 80)
35	79	80	(60 , 80)
47	84	85	(60 , 85)
50	89	90	(65 , 90)
51	94		

ROUTE 3 9.64 kms 65 min

ARRET	ARRIVEE	DEPART	FENETRE
51		41	
12	45	47	(15 , 45)
1	51	62	(60 , 105)
55	66	67	(60 , 120)
92	72	73	(60 , 90)
83	77	78	(60 , 85)
52	81	82	(30 , 120)
15	85	86	(60 , 90)
24	89	90	(70 , 90)
130	94	95	(60 , 120)
22	100	101	(60 , 120)
51	106		

ROUTE 4 4.46 kms 20 min

ARRET	ARRIVEE	DEPART	FENETRE
51		49	
53	55	56	(5 , 55)
17	60	61	(45, 95)
19	65	66	(50, 70)
51	69		

DISTANCE TOTALE : 45.47 kms

TEMPS TOTAL : 251 min.*

* incluant 28 min. d'attente

Fig. 5.6 (b) Algorithme PAR1

Chapitre 6. Extensions et développements futurs

Tel que nous l'avons indiqué, le système actuel se veut un outil exploratoire menant éventuellement à la production d'un S.E. oeuvrant dans le domaine de la génération de tournées et d'horaires pour des véhicules de transport. Il demeure toutefois possible d'envisager des modifications susceptibles d'étendre encore davantage le champ d'application du système actuel sans qu'il soit nécessaire d'y intégrer un savoir expert. De telles modifications supposent une utilisation du système en tant qu'outil privilégié pour le développement d'algorithmes de résolution mieux adaptés aux problèmes rencontrés (telle l'application décrite au chapitre 5). Par ailleurs, des extensions plus fondamentales verront à incorporer le savoir nécessaire à la production d'un système capable de se substituer dans une certaine mesure au concepteur d'algorithmes.

6.1 Les extensions ne modifiant pas la vocation actuelle du système

Les suggestions proposées ici représentent des modifications relativement simples à implanter, en ce sens qu'elles s'intègrent assez facilement au sein du modèle fourni par l'heuristique générale.

6.1.1 L'introduction de nouveaux opérateurs de base

Actuellement, l'ensemble des opérateurs de base assurant la modélisation de l'heuristique générale se limite à trois éléments, soit: INIT pour l'initialisation des routes, AJOUT pour l'insertion de nouveaux arrêts et ECHANGE pour la modification de la séquence des arrêts à l'intérieur des routes. L'introduction de nouveaux opérateurs au sein de cet ensemble aurait pour effet de multiplier les stratégies de résolution accessibles à l'usager. Nous pensons ici en particulier à deux opérateurs: FUSION qui permettrait de fondre deux ou plusieurs routes en une seule et SCISSION qui produirait l'effet contraire. L'introduction d'un opérateur de type SCISSION assurerait par exemple la reproduction des algorithmes utilisant une approche "route first, cluster second" où une route unique non réalisable est d'abord générée avant d'être scindée en un certain nombre de sous-routes réalisables.

Bien sûr, l'addition de tels opérateurs pose des problèmes liés à leur définition exacte. Ainsi, lors de l'application d'un operateur de type FUSION, selon quels critères doit-on réunir les routes? S'agit-il simplement de les concaténer (mettre bout-à-bout) ou faut-il au contraire recourir à des stratégies plus subtiles afin d'assurer un ordonnancement adéquat des noeuds au sein de la nouvelle route? Ces questions sont évidemment fort importantes et devront être considérées explicitement lors de la conception de ces opérateurs.

6.1.2 La résolution de problèmes comptant plusieurs origines

L'heuristique générale, dans sa forme actuelle, s'attaque à des problèmes ne comptant qu'une seule origine. Il serait sans doute bienvenu de modifier cette heuristique afin de lui permettre de traiter des problèmes avec plusieurs origines. Une telle extension devrait d'ailleurs se faire avec facilité car les concepts à la base de la construction des routes demeurent inchangés. De fait, la seule difficulté majeure tient à l'opérateur ECHANGE qui devra assurer le transfert des arrêts entre deux véhicules affectés à des origines différentes. Actuellement, cet opérateur ne permet en effet qu'un transfert entre deux véhicules affectés à une même origine (cf. le recours à l'équivalence MTSP - TSP).

6.1.3 La résolution de problèmes apparentés

Les concepts utilisés par l'heuristique générale sont susceptibles de s'appliquer à d'autres problèmes touchant à la génération de routes dans un réseau. On pense en particulier à certains problèmes de répartition ou de transport sur demande. La difficulté majeure provient du fait que ce sont souvent des problèmes de cueillette et de livraison qui supposent des contraintes de précédence entre les différents points de demande (il faut recueillir la demande au point A avant de la livrer au point B). L'heuristique générale n'a pas été conçue dans cette perspective et plusieurs concepts devront être repensés afin de l'adapter à de tels problèmes.

Il apparaît enfin qu'une nouvelle version du programme pour la résolution de problèmes où l'on dessert des arcs serait la bienvenue ("arc routing problems"). L'adaptation de l'heuristique générale à un tel contexte pourrait d'ailleurs se faire sans trop de difficultés. La modification majeure consiste à spécialiser la classe ARC plutôt que la classe NOEUD, puisque ce sont maintenant les arcs qui sont desservis par les véhicules. Evidemment, les méthodes qui assurent le traitement des arrêts dans notre implantation devraient être modifiées de façon à traiter des arcs. Il reste que les principes généraux sur lesquels repose la construction des routes demeurent les mêmes et la définition des opérateurs de base ne devrait donc pas nécessiter de modifications majeures.

6.1.4 La génération automatique de programmes

Comme nous l'avons mentionné au chapitre 4, il existe un mécanisme permettant de conserver des instanciations complètes des opérateurs INIT, AJOUT et ECHANGE. Puisqu'un programme se définit essentiellement comme une suite d'opérateurs correctement instanciés, nous disposons dès lors de tous les éléments assurant la génération automatique de programmes. On peut cependant douter de l'utilité véritable d'une telle approche. Il serait probablement plus

intéressant de chercher à générer des programmes en FORTRAN ou en PASCAL, qui sont des langages plus couramment utilisés par les chercheurs en R.O.

Ajoutons de plus que l'efficacité de ces programmes pourrait laisser à désirer. En effet, comme ces programmes doivent se mouler à une structure de base générale, il est impossible de recourir à certains "trucs de programmation" afin de réduire le temps d'exécution pour une application particulière. Il faudrait donc prévoir en pratique une intervention manuelle de façon à assurer la mise-au-point de programmes plus performants.

6.1.5 L'implantation de l'heuristique générale sur de nouveaux supports

Il pourrait être intéressant d'implanter le programme sur des supports moins spécialisés que la machine Lisp. Nous pensons particulièrement à une version pour micro-ordinateurs ainsi qu'à une version pour une machine d'utilisation plus générale de type SUN. Le transfert du programme sur une autre machine impliquerait bien sûr des modifications importantes et entraînerait certaines pertes touchant l'intégration des facilités graphiques et la représentation des réseaux sous forme d'objets symboliques. La machine Lisp offre en particulier un environnement privilégié pour les expérimentations les plus diverses. Elle permet d'identifier rapidement les approches de résolution les plus prometteuses et donc les plus susceptibles d'être implantées sur des machines offrant moins de flexibilité.

Il reste néanmoins que les concepts à la base de l'heuristique générale ne sont liés à aucune machine particulière, si ce n'est qu'il est souhaitable de compter sur un minimum d'interaction entre l'usager et le système. Il serait même possible d'en récolter certains bénéfices au niveau des temps d'exécution puisque la machine Lisp se révèle assez peu performante lors de calculs numériques.

6.2 Les extensions visant à produire un système expert

Les développements susceptibles de mener à la production d'un S.E. oeuvrant dans le domaine de la génération de tournées et d'horaires se divisent en cinq phases principales, soit:

1) La conception d'une classification générale

   Le travail visant à assurer une classification générale des divers types de problèmes rencontrés est présentement en voie de réalisation [6]. Une telle classification permettra de mieux situer chacun des problèmes en regard de certains critères bien définis, comme la nature des points de demande à desservir,

la nature des véhicules disponibles, les contraintes présentes, les objectifs à atteindre, etc.

2) L'identification des caractéristiques fondamentales des problèmes traités

Le travail de classification mentionné au point 1), devrait faciliter l'identification des caractéristiques susceptibles d'influencer les stratégies de résolution à adopter. On pense par exemple au degré d'élasticité des contraintes présentes, au nombre de points de demande à desservir, au nombre maximal de véhicules disponibles, etc.

3) La conception d'un modèle général permettant la spécification des stratégies de résolution appropriées

C'est ici le sujet fondamental de cette thèse, et l'heuristique générale que nous avons développée permet effectivement de reproduire une classe importante d'algorithmes de résolution. Il va de soi que ce modèle est susceptible d'être augmenté ou modifié en fonction des besoins nouveaux qui peuvent survenir. Néanmoins, la présente thèse a permis de démontrer que l'intégration d'un tel modèle général à l'intérieur d'un système informatisé est en mesure d'apporter des bénéfices appréciables.

4) La déduction automatique des stratégies de résolution en fonction des caractéristiques des problèmes traités

Cette phase de développement apparaît assez délicate à cause de la multiplicité et de la diversité des contraintes présentes au sein des problèmes rencontrés. Le système que nous avons conçu devrait toutefois permettre d'accélérer l'acquisition de cette expertise en assurant l'expérimentation des approches de résolution les plus diverses à l'aide d'une simple modification des formules d'instanciation de l'heuristique générale.

Bien sûr, il faudra par la suite intégrer ce savoir à l'intérieur du système et développer un mécanisme d'inférence approprié. Cette phase du travail correspond de fait à l'implantation pratique des idées et concepts recueillis au cours des phases précédentes. Certaines techniques de manipulation symbolique, propres à l'intelligence artificielle, devront ici être utilisées afin d'assurer l'automatisation des processus de décision lors de l'instanciation de l'heuristique générale. L'intégration future des connaissances nécessaires à la réalisation d'un

véritable S.E. explique d'ailleurs le recours à l'approche "orientée objet" lors de la conception de notre système.

5) L'intégration de facilités assurant une modification interactive de la solution

La phase finale de développement verra a concevoir des facilités permettant à l'usager de modifier interactivement la solution produite, de façon à répondre à certains besoins spécifiques liés à une application particulière.

La réalisation de toutes les étapes mentionnées ci-haut assurerait alors la production d'un système capable de concevoir automatiquement des algorithmes de résolution adaptés aux différents types de problèmes de génération de tournées et d'horaires. Ce système pourrait ainsi se substituer dans une certaine mesure aux experts chargés de la conception de ces algorithmes.

Conclusion

La présente thèse nous a permis de définir une heuristique générale grâce à laquelle il est possible de concevoir et expérimenter diverses stratégies de résolution face à des problèmes complexes de génération de tournées. Cette heuristique générale est elle-même intégrée au sein d'un système graphique et interactif qui permet à l'usager de construire dynamiquement les routes désirées.

L'intérêt de notre système tient au fait qu'il permet d'explorer rapidement de nouvelles approches de résolution face à des problèmes pour lesquels aucun algorithme connu n'apparaît vraiment satisfaisant. L'approche retenue permet ainsi de "briser" la barrière de l'inconnu et accélérer le développement de nouveaux algorithmes heuristiques mieux adaptés aux problèmes qui confrontent les chercheurs oeuvrant dans ce domaine. Notre système a ainsi permis d'identifier des solutions de qualité supérieure à celle d'un algorithme spécialisé pour un problème complexe de collecte du courrier.

ALTO s'est également avéré utile afin d'illustrer le fonctionnement de certains algorithmes classiques utilisés lors de la génération de tournées. A cet effet, les facilités graphiques du système ont été mises à profit à l'occasion de travaux pratiques pour le cours IFT6750 (Tournées et Horaires: modèles, algorithmes).

Bien sûr, de nombreuses extensions au système actuel sont envisagées afin d'étendre encore davantage son champ d'application. Nous pensons ici en particulier à l'introduction de contraintes de précédence assurant le traitement de problèmes mixtes de collecte et de livraison ainsi que l'introduction de contraintes touchant au transport de commodités multiples. De plus, de nouveaux opérateurs de base, tel un opérateur assurant la fusion des routes, seront intégrés au système afin d'enrichir l'heuristique générale et multiplier les stratégies de résolution disponibles à l'usager.

Nous espérons enfin que notre système servira de point de départ à la conception d'un véritable S.E. capable de suggérer automatiquement des algorithmes de résolution bien adaptés aux problèmes soumis.

Annexe

A la fin du chapitre 4, nous avons présenté des exemples d'algorithmes qui peuvent être reproduits grâce au modèle offert par l'heuristique générale. Les formules permettant d'instancier le modèle était toutefois représentées à l'aide d'une notation abrégée, de nature symbolique. Nous reprenons donc ici les mêmes exemples en y introduisant cette fois la notation réelle en CLISP (Conversational Lisp).

Dans tous les exemples qui suivent, nous avons :

a) une origine o

b) une flotte homogène

c) $k' \in SE_k$; $i' \in E_i$; $s' \in E_s$; $r' \in E_r$;

d) $j' = SUCC_{i'}$

Exemple 1. "Farthest insertion"

INIT

N ← 1;

SE_k ← E_k;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← **(FOR k' IN SE_k MAXOBJ (DIST o k'))**;

v ← **(RANDOBJ $E_{v|}$)** ;

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N-1;

AJOUT

SE_k ← E_k

Tant que SE_k ≠ ∅ faire

r_1 ← **NIL**;

k ← **(FOR k' IN SE_k MAXOBJ (FOR i' IN E_i MINVAL (DIST i' k')))**;

r_2 ← **NIL**;

i ← **(FOR i' IN E_i MINOBJ (DETOUR i' k j'))**;

si **NIL** vérifié alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k après i

sinon

traiter k selon **mode libre**;

retirer k de SE_k;

Contraintes : aucune

Exemple 2 "Nearest neighbor"

INIT

N ← **1**;

SE_k ← E_k;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← **(FOR k' IN** SE_k **MINOBJ (DIST o k'))**;

v ← **(RANDOBJ** E_{vl}**)**;

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N - 1;

AJOUT

SE_k ← E_k;

Tant que $SE_k \neq \emptyset$ faire

r_1 ← **(NOUV** E_r**)**;

k ← **(FOR k' IN** SE_k **MINOBJ (DIST (DERNIER** r_1**) k'))**;

r_2 ← **NIL**;

i ← **(DERNIER** r_1**)**;

si **NIL** vérifié alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k **après** i

sinon

traiter k selon **mode libre**;

retirer k de SE_k;

Contraintes : aucune

Exemple 3. Algorithme de balayage

On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'il n'y ait plus d'arrêts libres.

INIT

N ← 1;

SE_k ← E_k;

Tant que N > 0 et qu'il reste des véhicules disponibles faire

GERMES ← (if (VIDE? E_r) then
(FOR k' IN SE_k MAXOBJ (DIST o k'))
else
(FOR k' IN SE_k MINOBJ
(FOR s' IN E_S MINVAL (ANGLEH s' o k'))));

v ← (RANDOBJ E_{vl}) ;

créer une route où les GERMES sont desservis par le véhicule v;

SE_k ← SE_k - GERMES;

N ← N-1;

AJOUT

SE_k ← E_k;

Tant que SE_k ≠ Ø faire

r_1 ← (NOUV E_r);

k ← (FOR k' IN SE_k MINOBJ (ANGLEH (DERNIER r_1) o k'));

r_2 ← NIL;

i ← (DERNIER r_1);

si NIL vérifié alors sortir de la boucle et terminer;

si l'insertion de k respecte les contraintes alors

insérer k après i

sinon

traiter k selon mode interruption;

retirer k de SE_k;

Contraintes : LAMBDAcap = 1.0

Exemple 4. Algorithme de Ferland et Rousseau

Phase 1. On active successivement les opérateurs INIT et AJOUT jusqu'à ce qu'on ait ⌈demande totale / capacité⌉ routes

INIT

$N \leftarrow 1$;
$SE_k \leftarrow E_k$;
Tant que N > 0 et qu'il reste des véhicules disponibles faire
    GERMES ← **(FOR k' IN SE_k MAXOBJ (DIST k' o))**;
    v ← **(RANDOBJ E_{vl})** ;
    créer une route où les GERMES sont desservis par le véhicule v;
    SE_k ← SE_k - GERMES;
    N ← N-1;

AJOUT

$SE_k \leftarrow E_k$;
Tant que $SE_k \neq \emptyset$ faire
    r_1 ← **(NOUV E_r)**;
    k ← **(FOR k' IN SE_k MINOBJ** (formule de Ferland & Rousseau)
        (FOR s' IN E_s MINVAL
            (if (DIST s' k') LE (DIST s' o) then
                (DIST s' k') ↑ 2 * (1 + LAMBDA *
                (1 - ((DIST s' k') ↑ 2 - (DIST k' o) ↑ 2 + (DIST s' o) ↑ 2) /
                (2 * (DIST s' k') * (DIST s' o))))
            else (DIST s' k') ↑ 2 * (1 + LAMBDA * ((DIST k' o) - (DIST s' k'))))));
    r_2 ← **NIL**;
    i ← **(RANDOBJ E_i)**;
    si **NIL** verifié alors sortir de la boucle et terminer;
    si l'insertion de k respecte les contraintes alors
        insérer k **après** i
    sinon
        traiter k selon **mode interruption**;
    retirer k de SE_k;

Contraintes : LAMBDAcap = **1.0**

Exemple 4. Algorithme de Ferland et Rousseau

Phase 2. On active l'opérateur AJOUT de façon à introduire les arrêts qui demeurent libres suite à la phase 1.

AJOUT

```
SEk ← Ek;
Tant que SEk ≠ Ø faire
        r1 ← (FOR r' IN Er MINOBJ (DEMtot r'));
        k  ← (FOR k' IN SEk MINOBJ
                (FOR i' IN Ei MINVAL
                  (if (DIST i' k') LE (DIST i' o) then
                        (DIST i' k') ↑2 * (1 + LAMBDA *
                          ( 1 - ((DIST i' k') ↑2 - (DIST k' o) ↑2 + (DIST i' o) ↑2) /
                                                    (2 * (DIST i' k') * (DIST i' o))))
                    else  (DIST i' k') ↑2 * (1 + LAMBDA * ((DIST k' o) / (DIST i' k'))))));
        r2 ← NIL;
        i  ← (RANDOBJ Ei);
        si NIL vérifié alors sortir de la boucle et terminer;
        si l'insertion de k respecte les contraintes alors
                insérer k après i
        sinon
                traiter k selon mode libre;
        retirer k de SEk;
```

Remerciements

J'aimerais remercier ici M. Guy Lapalme, professeur agrégé au département d'informatique et de recherche opérationnelle, pour sa très grande disponibilité, ses conseils et son soutien tout au long de cette recherche. J'aimerais également remercier M. Jean-Marc Rousseau, directeur du Centre de Recherche sur les Transports, pour les idées et les conseils qui ont permis d'assurer la progression de mon travail. Enfin, ce projet a pu être mené à terme grâce au support financier de M. Jacques Ferland, professeur titulaire au département d'informatique et de recherche opérationnelle, et grâce aux bourses offertes par le C.R.S.N.G. et le C.R.T.

Bibliographie

[1] Aikins J.S., "Prototypical Knowledge for Expert Systems", Artificial Intelligence 20, pp 163-210 (1983).

[2] Bel G., Bensana E. & Dubois D., "Un Système d'Ordonnancement Prévisionnel d'Atelier utilisant des Connaissances Théoriques et Pratiques", 6e Journées Inter., "les S.E. et leurs applications", Avignon, pp 757-770 (1986)

[3] Bellmore A. & Hong S., "Transformation of Multi-Salesman Problem to the Standard Traveling Salesman Problem", J. ACM, pp. 500-504 (1974).

[4] Bodin L., Golden B., Assad A. & Ball M., "Routing and Scheduling of Vehicles and Crews: The State of the Art", Computer and Operations Research, vol. 10, no. 2 (1983).

[5] Cullen F., Jarvis J. & Ratliff D., "Set Partitioning Based Heuristics for Interactive Routing", Networks, vol. 11, pp. 125-143 (1981).

[6] Davis R. & Lenat D., Knowledge Based Systems in Artificial Intelligence, McGraw-Hill, NY (1982)

[7] Desrochers M., Communication privée

[8] Desrochers M., "Les Systèmes Experts et la Recherche Opérationnelle", Publication no. 386, CRT, Université de Montréal (1984)

[9] Desrosiers J., Ferland J.A., Rousseau J.M., Lapalme G. & Chapleau L. "TRANSCOL: A Multi-Period School Bus Routing and Scheduling System", TIMS Studies in the Management Sciences 22, pp. 47-71 (1986).

[10] Elliott R.J. & Lesk M.E., "Route finding in Street Map by Computer and People", Proc. AAAI, pp. 258-261 (1982).

[11] Ferland J.A., Rousseau J.M. & Chapleau L., "Clustering for Routing in Densely Populated Areas", European Journal of Operational Research 20, pp. 48-57 (1985).

[12] Fikes R.E. & Nilsson N.J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, vol. 2, pp. 189-208 (1971).

[13] Fisher M. & Jaikumar R., "A Generalized Assignment Heuristic for Vehicle Routing", Networks, vol. 11, pp. 109-124 (1981).

[14] Fox M.S., "Constraint-Directed Search: A Case Study of Job-Shop Scheduling", Ph. D. Thesis, Carnegie-Mellon University, Pittsburgh, Pensylvania (1983).

[15] Fox M.S., "Observation on the Role of Constraints in Problem Solving", Proc. of the sixth Canadian Conf. on Artificial Intelligence, pp. 172-187 (1986).

[16] Fukumori K., "Fundamental Scheme for Train Scheduling", MIT AI Memo no. 596, Artificial Intelligence Laboratory, MIT, Cambridge, MA (1980).

[17] Gillett B. & Miller L., "A Heuristic Algorithm for the Vehicle Dispatch Problem", Operations Research 22, pp. 340-349 (1974).

[18] Golden B., "Evaluating a Sequential Vehicle Routing Algorithm", AIIE Trans. 9, pp. 204-208 (1977).

[19] Goldstein I.P., "Bargaining between Goals", IJCAI 4, pp. 175-180 (1975).

[20] Goldstein I.P. & Roberts R.B , "NUDGE, a Knowledge Based Scheduling Program", IJCAI 5, pp. 257-263 (1977).

[21] Hayes-Roth F., Waterman D.A. & Lenat D B , Building Expert Systems, Addison-Wesley (1983).

[22] Jacquet-Lagreze E., "PREFCALC: Evaluation et Décision Multicritères", manuel de l'usager, EURO-DECISION eds. (1983).

[23] Krolak P. & Felts W., "A Man-Machine Approach toward Solving the Traveling Salesman Problem", Communications of the ACM, vol. 4, nol. 5, pp. 327-334 (1971).

[24] Krolak P., Felts W. & Nelson J., "A Man-Machine Approach toward Solving the Generalized Truck-Dispatching Problem", Transportation Science 6, pp. 149-170 (1972).

[25] Laurière J.L., "A Language and a Program for Stating and Solving Combinatorial Problems", Artificial Intelligence 10, pp. 29-127 (1978).

[26] Laurière J.L., "Un Langage Déclaratif: SNARK", TSI, vol. 5, no. 3, pp. 141-172 (1986).

[27] Lin S., "Computer Solutions of the Traveling Salesman Problem", Bell System Techn. J. 44, pp. 2245-2269 (1965).

[28] Little J.D , Murty K.G. Sweeney D.W., & Karel C., "An Algorithm for Traveling Salesman Problem", Operations Research 11, pp. 972-989 (1963)

[29] Michie D., Fleming J.G. & Oldfield J.V., "A Comparison of Heuristic, Interactive and Unaided Methods of Solving a Shortest-Route Problem", Machine Intelligence, D. Michie eds., American Elsevier, New York, pp. 245-252 (1968).

[30] Minsky M., "A Framework for Representing Knowledge", in The Psychology of Computer Vision, P. Winston eds., McGraw-Hill, NY (1975).

[31] O'Keefe R., "Expert Systems and Operational Research - Mutual Benefits", J. Op. Res. Soc., vol. 36, no. 2, pp. 125-129 (1985).

[32] Roche C., "EAQUE-LRO. Génération de Systèmes Experts. Application à des Problèmes d'Ordonnancement", Thèse de troisième cycle, Institut National Polytechnique de Grenoble (1984).

[33] Rosenkrantz D., Sterns R. & Lewis P., "An Analysis of Several Heuristics for the Traveling Salesman Problem", SIAM J. Comp. 6, pp. 563-581 (1977).

[34] Rousseau J.M. & Blais J.Y., "HASTUS: An Interactive System for Buses and Crew Scheduling", in Computer Scheduling of Public Transport 2, J.M. Rousseau eds, North-Holland, pp. 45-60 (1985).

[35] Sacerdoti E.D., "Planning in a Hierarchy of Abstraction Spaces", Artificial Intelligence, vol. 5, no. 2, (1974).

[36] Sacerdoti E.D., "The Non Linear Nature of Plans", IJCAI 4, pp. 206-214 (1975).

[37] Shapiro D., "Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem", Sc. D. Th., Washington University, St-Louis (1966).

[38] Stefik M., "Planning with Constraints (MOLGEN: Part 1)", Artificial Intelligence 16, pp. 111-140 (1981).

[39] Stefik M., "Planning and Meta Planning (MOLGEN: Part 2)", Artificial Intelligence 16, pp. 141-170 (1981).

[40] Stefik M. & al., "The Organization of Expert Systems: a Tutorial", Artificial Intelligence 18, pp. 135-173 (1982).

[41] Stefik M., Bobrow D., Mittal S. & Conway L., "Knowledge Programming in Loops", The AI Magazine, pp. 3-13 (1983).

[42] Tate A., "Generating Project Networks", IJCAI 5, pp. 888-893 (1977).

[43] Teitelman W. & Masinter L., "The Interlisp Programming Environment", Computer, pp. 25-32 (1981).

[44] Tuan P.L., "Current Applications of Advanced Technologies to Transportation Operations", Proc. of the Workshop on the Application of Expert Systems to Transportation, TRANSPORTS CANADA eds, section III, pp. 1-7 (1985).

[45] Winograd T., "Frame Representation and the Declarative-Procedural Controversy", in Representation and Understanding, D.G. Bobrow & A. Collins eds., Academic Press, NY (1975).

[46] Xerox Corp., Interlisp Reference Manual (1986).